$2.00

Personal Computing

MAY/JUNE, 1977

Undercover Computer

Robots On Your Doorstep

A Lemonade Vision

Barry Goldwater Jr. On Privacy

The Personal Software Genie

Meet the most powerful μC system available for dedicated work. Yet it's only $595.*

Here's the muscle you've been telling us you wanted: a powerful Cromemco microcomputer in a style and price range ideal for your dedicated computer jobs—ideal for industrial, business, instrumentation and similar applications.

It's the new Cromemco Z-2 Computer System. Here's some of what you get in the Z-2 for only $595:

- The industry's fastest μP board (Cromemco's highly regarded 4 MHz, 250-nanosecond cycle time board).
- The power and convenience of the well-known Z-80 μP.
- A power supply you won't believe (+8V @ 30A, +18V and −18V @ 15A — ample power for additional peripherals such as floppy disk drives).
- A full-length *shielded* motherboard with 21 card slots.
- Power-on-jump circuitry to begin automatic program execution when power is turned on.
- S-100 bus.
- Standard rack-mount style construction.
- All-metal chassis and dust case.
- 110- or 220-volt operation.

DEDICATED APPLICATIONS

The new Z-2 is specifically designed as a powerful but economical dedicated computer for systems work. Notice that the front panel is entirely free of controls or switches of any kind. That makes the Z-2 virtually tamper-proof. No accidental program changes or surprise memory erasures.

FASTEST, MOST POWERFUL μC

Cromemco's microcomputers are the fastest and most powerful available. They use the Z-80 microprocessor which is widely regarded as the standard of the future. So you're in the technical fore with the Z-2.

BROAD SOFTWARE/PERIPHERALS SUPPORT

Since the Z-2 uses the Z-80, your present 8080 software can be used with the Z-2. Also, Cromemco offers broad software support including a monitor, assembler, and a BASIC interpreter.

The Z-2 uses the S-100 bus which is supported by the peripherals of dozens of manufacturers. Naturally, all Cromemco peripherals such as our 7-channel A/D and D/A converter, our well-known BYTESAVER with its built-in PROM programmer, our color graphics interface, etc., will also plug into the S-100 bus.

LOW, LOW PRICE

You'll be impressed with the Z-2's low price, technical excellence and quality. So see it right away at your computer store—or order directly from the factory.

Z-2 COMPUTER SYSTEM KIT (MODEL Z-2K) (includes 4 MHz μP card, full-length 21-card-slot motherboard, power supply, one card socket and card-guide set, and front panel; for rack mounting) . . . .$595.

Z-2 COMPUTER SYSTEM ASSEMBLED (MODEL Z-2W) (includes the above as well as all 21 sockets and card guides and a cooling fan; for rack mounting) . . .$995.

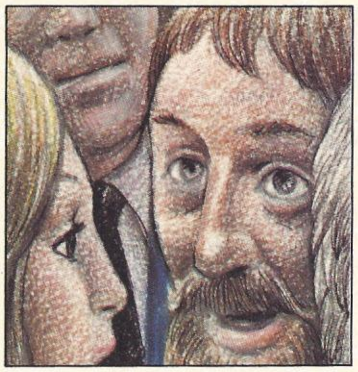
Shown with optional bench cabinet

*kit price

Cromemco
incorporated
Specialists in computers and peripherals
2432 CHARLESTON RD., MOUNTAIN VIEW, CA 94043 • (415) 964-7400

CIRCLE 4

48



Another Lemonade Computer Service that combines profits with service for national or neighborhood shindigs.

66

Tom Munnecke shows you how easy it can be to talk to your computer.

104

Rick Loomis' crystal ball tells you all about the future of computer gaming.

140

Aldo Giorgini's conceptual extension of our star spangled banner.

SUBSCRIPTION POLICY

Subscription rates: Six issues — U.S., $8; Canada and Mexico add $2 for surface mail and $4 for airmail; all other countries add $4 for surface mail and $18 for airmail. Twelve issues — U.S., $14; outside U.S., postage rates doubled. Back issues (as available): $2 per copy in U.S.; $4 in Canada and Mexico; $5 in all other countries. Send subscription orders and changes of address to: Circulation, Personal Computing, 167 Corey Rd., Brookline, MA 02146.

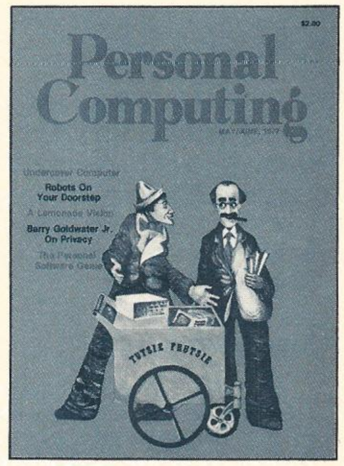
Personal Computing is published bimonthly by Benwill Publishing Corp., 167 Corey Road, Brookline, MA 02146. Editorial office: 401 Louisiana S.E., Alburquerque, NM 87108; telephone (505) 266-1173 Publication office: 167 Corey Road, Brookline, MA 02146; telephone (617) 232-5470. Application to mail at controlled circulation rates is pending at Long Prairie, MN. Membership application in Audit Bureau of Circulation pending.

CONTENTS

Personal Computing

MAY/JUNE 1977 VOL. 1 NO. 3

COVER



Artist Don Nace's illustration of what Chico would have sold to Groucho if personal computing had been in Vogue during "A Day at the Races".

All editorial correspondence should be addressed to PERSONAL COMPUTING, 401 Louisiana S.E., Alburquerque, NM 87108.

We welcome submission of manuscripts for publication and pay competitive rates for material accepted. Authors should study the magazine for content and style to avoid sending inappropriate material at expense in time and postage.

Always include a stamped, self-addressed envelope if you want material returned. PERSONAL COMPUTING takes no responsibility for the safekeeping of materials not specifically solicited but attempts to respond promptly to submissions.

STAFF

Letters

Gentlemen:

. . . I must disagree with your editorial suggesting the end of the kit. You mentioned 85 to $110 per month service contracts. When you build your own, you service your own. So you save $1020 to $1320 per year to buy extra equipment. . .

Jack H. Peterson
Saginaw, Michigan

That $85 monthly service contract figure was part of an interview with a chap who has an IBM 5100. He's paying that – and so might any of us if we could afford it – not only for miscellaneous repair, but for technical updating of a ten thousand dollar system. IBM isn't a cut-rate house and their service is superb. This is not typical of personal computing equipment and rates, though we have never seen anybody else's firm contract figure for maintenance of a computer/tape deck/CRT system. And yes, if you build your own kit you can probably service it, but that is like observing that "You can say anything *in French – if only you know how." Kits are great for kit builders, of whom there are many, representing a continuing special market.*

To the Editor,

. . . To quote a recent Postal Service study: The United States Postal Service should explore fully and agressively roles in future transaction and communication systems. There may be a role for USPS to operate a tele-communications switching system.

In my opinion, the last thing computer hobbyists and manufacturers need is to have our tax dollars subsidize a "Postal Service" which forces us to purchase Government standardized terminals and peripheral equipment to plug into a Government run telecommunication system to send, for all we know, Government approved hard copy.

Rebecca Coffey
San Francisco, CA

Dear Mr. Winkless,

. . . Henry Gilroy's "ten easy steps to becoming a computer hobbyist" (January/February issue) valiantly succeeds in seducing the neophyte into computing. However, many of his generalized steps need to be elaborated in future issues. . . For example, in Step Five, Gilroy advises evaluating present-day equipment either indirectly via vendor's literature or directly via local retailers. . . . I question whether this newcomer, outfitted with his inadequate (if not totally deprived) technological background, could intelligently decipher any manufacturer's data or could choose between an Altair and an IMSAI at a store which is biased in favour of its own supplier.

. . . I have one major complaint – you are one of the very rare publishers who insist on ridiculous subscription rates for Canadian subscribers. (Personal Computing rates for Canadians are almost double US rates!)

Walter V. Plawan
N. Vancouver, B.C.

Indeed, we must not only expand the material, but repeat it as time passes to be useful to an expanding population of computer-buyers. You'll find more material issue by issue. And you're right about ridiculous surcharges for Canadian subscribers. You'll see changes in coming months. No slight intended. We have many Canadian subscribers and we will do better by them.

Re: **An Informal History of the Computer Market** by Alan R. Kaplan Personal Computing Jan/Feb 1977

Sir,

I recently had the opportunity to read your Premier Issue. I found it to be highly polished, informative, and fun to read!

Best wishes for continued success.

Morey Schapira, Editor
THE HARBUS NEWS
Student Newspaper of the
Harvard Business School

Dear Alan,

Thanx for the nice words in Personal Computing. PCC (newspaper) was first published Oct. 1972. But actually, I really started in 1962. By 1972 we already had a course going thru University of California Extension called "Games Computers Play." *My Computer Likes Me* came out in 1972 at an original price of $1.19; it is now $2.00. (Inflation!)

Alan, I saw no mention of Hal Singer and the *Micro-8* newsletter & user's group. *Micro-8* is the real forerunner of all the current "hobby" periodicals and Hal Singer should get heaps of praise!

Bob Albrecht
Menlo Park, CA

The author replies:

I thank Mr. Albrecht for setting me straight on the original publication date and price of My Computer Likes Me When I Speak in BASIC. *His comments regarding Hal Singer's contributions are also accurate. I have never had the pleasure of meeting Mr. Albrecht personally, but his desire to minimize his own contributions and to share credit with others would appear to be characteristic of him.*

Alan R. Kaplan
Venture Development Corp.

Dear Sir:

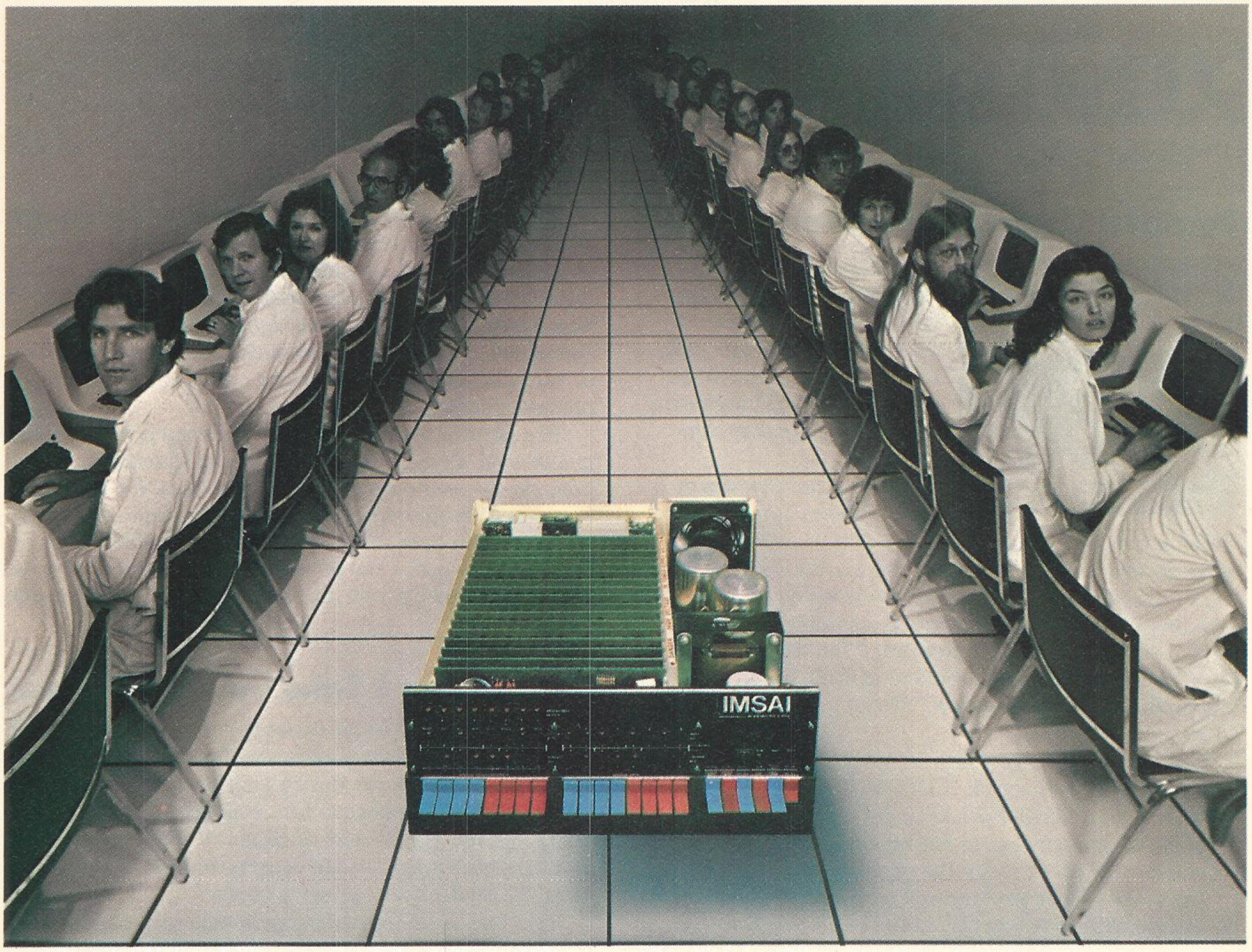
After making the necessary conversion, I tried your "Story" game (page 79, January/February) on the IBM 5100. I found, however, that the randomize instruction would always yield the value '3' for V, L, and T. Thus, we always went to the "DESERT" on a "PIG" and found a "SHACK". I then modified statements 5070, 5071, and 5072 and inserted three new line of instructions as follows:

```
5070 V=INT(3*RND(A)+1)
   where you listed. . .(3*RND(0). . .
5071 L=INT(3*RND(B)+1)
   where you listed. . .(3*RND(0). . .
5072 T=INT(3*RND(C)+1)
   where you listed. . .(3*RND(0). . .
           A=V
inserted   B=L
           C=T
```

The program then produced the various versions of the story in a random manner.

John N. Bishara
Director Information Systems
The Cafaro Company
Youngstown, OH

POWER.



IMSAI Introduces the Megabyte Micro.™

The Megabyte Memory

Until today, the largest memory you could fit and address in a single microcomputer CPU was 65K.

Now, IMSAI presents an incredible memory system for micros 16 times more powerful than yesterday's best.

Imagine, a full megabyte of power from sixteen 65K RAM boards.

And, to control all this, the IMSAI Intelligent Memory Manager (IMM), the super control board.

You can write protect blocks throughout the full megabyte. Or, map in 16K blocks.

Plus, preset 16 mapping configurations with protect for high speed transfer or rapid change.

All interrupts are fully vectored, and there's an interrupt if an attempt is made to write into protected memory.

There's even a real "time of day" clock.

65K, 32K and 16K RAM Boards

Until today, the most memory you could plug into a single slot was 16K.

Now, IMSAI presents memory boards in astonishing multiples of sixteen: 65K, 32K and 16K low power, dynamic RAM Boards. They can be used in any S-100 bus computer individually or in combination to form conventional systems up to 65K bytes.

Every board is fast. With "hidden refresh" and *no* "wait state."

The Complete Megabyte Microcomputer System

The IMSAI Megabyte Micro™ is only part of the story. The full system can include dual floppy disks, terminals, plotters, printers and tape cassettes.

IMSAI also offers the finest high level and peripheral software available. Paper tape and Tape Cassette I/O and super Disk Operating Systems. Plus, BASIC and Disk BASIC with more high level languages coming.

Until today, the microcomputer's potential was just something you talked about.

Now, you can put it to work. Powerfully.

GENTLEMEN: PC-56

I'm power hungry!

- ☐ Send 65K RAM Board Kit $2599 ☐ Assembled $3899
- ☐ Send 32K RAM Board Kit $749 ☐ Assembled $1099
- ☐ Send 16K RAM Board Kit $449 ☐ Assembled $679
- ☐ Send IMM ROM Control Kit $299 ☐ Assembled $399
- ☐ Send IMM EROM Control Kit $499 ☐ Assembled $699
- ☐ Send full catalog $1.00

Check/MO enclosed. Amt. $ ________

Charge my: ☐ BAC ☐ M/C

________ Exp. Date: ________

Sig. ________

☐ Send name of my nearest IMSAI dealer

Name ________

Company ________ Title ________

Address ________

City ________

State/Zip ________

IMSAI®

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093 TWX 910-366-7287

CIRCLE 5

MEMO FROM THE PUBLISHER

Personal Computing Abused

In March of 1976 at the World Altair Computer Convention in Albuquerque , a computer hobbyist stood up at one of the sessions to propose an intriguing business application for personal computing.

He proposed the idea of using a home computer to devise an automated phone solicitation machine. The computer-phone could systematically dial every number in a city or a section of a city. A taped voice would be controlled such that responses would be keyed according to certain speech recognition clues. If you said, "Yes, I'm interested in that," the computer would respond with a tempting sales pitch followed by the question, "Would you like to receive a free, full-color brochure?"

If you said "No," the computer would hang up and within microseconds ring another number. If you didn't answer the phone, your number would be put back into rotation. It could call you ten or even one hundred times before it became too inefficient to keep the gig going.

The proposal was not very well received. Lou Fields of the Southern California Computer Society said that if we wanted government interference in personal computing, this was one of the surest ways to bring it about. This type of business plan was insensitive to the rights of individual privacy. It could cause a public uproar.

Fortunately, the vast majority of the audience heartily agreed. Computer users in general are sensitive to this issue. They want to use computers in a responsible, constructive way. People at the WACC were glad Lou said what he did.

Will the U.S. Really Get Involved?

In England they are debating a law that would require all computer operators to pay an annual licensing fee. What are the chances of this type of law being proposed in the United States? (See *The Equalizer,* page 14).

It is my opinion that the chances are actually pretty good. Barry Goldwater Jr., has a bill before the House that would regulate all personal data files used by private institutions and individuals. While he says that he really hopes private concerns will regulate themselves and the law will therefore be unnecessary, chances of this happening seem slim. (See Goldwater interview, page 17).

The problem is partly just that lawmakers aren't aware of personal computing. They aren't aware that computers are about to become an exploding consumer product *at least* as big as handheld calculators. These laws are being written for large system users, universities, department stores, insurance companies, etc. One can suppose that once lawmakers learn of the imminent people/computer explosion, they'll make proper adjustments.

The question is do we dare tell them and if so, how much and in what way? Computers still get an abundance of bad press. Some people would like to strike back for the billing errors and the absurd lengths they've sometimes had to go to make corrections. They're mad and would like nothing better than to pass a law restricting the growth of computer technology. Some people are *still* afraid computers are going to take over.

The Reason I Take This So Seriously

I'm not trying to be an alarmist. For one thing, personal computers have already proliferated beyond most government regulation. People already have them, just like (pardon the analogy) people already have hand guns. . If you have a computer, use it. It is your equalizer. It is a way to organize and fight back against the impersonal institutions and the catch-22 regulations of modern society.

The issue of computers and personal privacy has been treated a hundred different ways by a hundred different publications. The reason we're expressing our concern in this issue is that the advent of personal computing changes *everything.* It is our hope that this new, personal technology can be used to reverse the mass alienation people feel towards society. Our concerns in this regard are similar to those expressed by Jules Bergman in our March/April edition.

A personal computer is something quite different from an institutional computer. The owners of personal computers need the freedom to develop their systems to increase personal power. Government regulations aimed at Insurance Companies should not apply to individual computer owners.

The reason I take this so seriously is because of an incident that occurred to me recently. I was sitting in the family room at my home watching TV. The phone rang. I answered. It was a recording. It wanted to know if I was interested in finding out about a land deal in southern Colorado. I said "NO!" It hung up.

David Bunnell

INTRODUCING THE COMPLETE CHALLENGER SYSTEM.

OSI's DREAM MACHINE
is a totally integrated computer system.

Imagine a system complete with terminal, CPU, memory, floppy disk, software, and all the little necessities to make it work together immediately. Now imagine this complete system available not only fully assembled, but priced much lower than anyone else's kit. What you are dreaming of is OSI's "new" Challenger System!

In the configuration shown above, the Challenger includes everything an end user needs for a complete small computer system. All you add is 110 VAC power and a desk to put it on.

This fully-assembled system includes:

HARDWARE:

OSI Challenger 65 with 16K RAM, serial interface, system monitor PROM, and floppy disk bootstrap PROM.

OSI Challenger single drive floppy disk formatted for 250K bytes storage per diskette surface.

Stand-alone terminal and Sanyo monitor for 16 lines of 64 characters at 2400 baud (other terminal options are available).

And all interconnecting cables!

SOFTWARE:

2 diskettes containing over 100,000 bytes of software including OSI's powerful Disk Operating System with variable length sectors. **6502 DISK BASED RESIDENT ASSEMBLER/EDITOR!** A totally interactive Assembler/Editor which assembles up to 600 lines a minute and is completely compatible with MOS Technology's Cross Assembler format. This program also contains a powerful disk-based line editor with commands for general text editing. **OSI'S EXTENDED MONITOR:** A powerful machine language debugging and utilities package including a Disassembler which is format compatible with the Assembler! **OSI 6502 8K BASIC FOR DISK BY MICROSOFT:** This powerful BASIC has all the features of Altair® 8K BASIC for the 8080 plus higher speed and disk storage. And it comes complete with a BASIC program library.

DOCUMENTATION AND SUPPORT:

We include over 600 pages of hardware, software, programming, and operation manuals. The Challenger is based on the well-proven OSI 400 system. The over 2,000 OSI 400s and Challengers now in use assure continuing hardware and software support for this system for years to come!

EXPANDABILITY:

The Challenger System can now be expanded to 192K of RAM and 16K of I/O and ROM. There are over 13 accessory boards including A/D, D/A, parallel and serial I/O, cassette interfaces, a dual drive floppy, a video graphics display, several RAM and PROM boards, and multiple-processor configurations.

APPLICATIONS:

The Challenger system is complete, fully assembled and configured so that the Disk Operating System can be booted in immediately on system power-up. Even a relatively inexperienced operator can have a complex BASIC program on-line just seconds after the system is turned on. The ease of use, high reliability, and large library of standard BASIC applications programs make the OSI Challenger System the first practical and affordable small computer system for small business, educational institutions, labs, and the personal computerist.

PRICES:

Challenger System, complete as stated above with terminal and monitor

$2599.00

As above without terminal. Specify RS-232 or 20ma loop and baud rate

$2099.00

IMPORTANT NOTE:

One of the most important features of the Challenger System is that it is not really "new". OSI has been delivering the basic circuitry of the Challenger since November 1975 and the floppy disk since June 1976. The only thing new is the total integration of the components as a complete, simple to use, fully-assembled, small computer system.

For more free information and the address of the OSI Computer Dealer or representative in your area, write to: OSI; Dept. S; Hiram, Ohio 44234 or enclose $1.00 for the full OSI catalog which contains kits from $134 and fully assembled computers from $439.

OSI

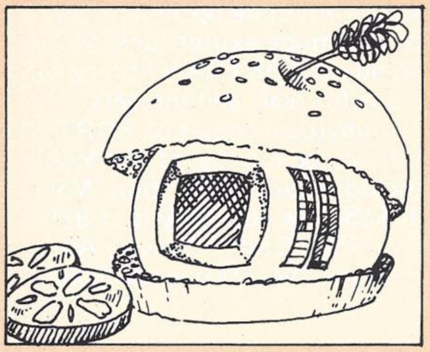
Ohio Scientific Instruments

11679 Hayden Street, Dept. S, Hiram, Ohio 44234

CIRCLE 7

Adam and Eve on a Raft

Computer retailers are usually so busy minding the store that they can't spend time to develop sales literature genuinely useful to their customers. Most local store catalogs are baffling and depressing lists packed into poorly printed pamphlets. Luckily, the COMPUTER CENTER, Inc., with stores in San Francisco and Berkeley, is offering a free DO-IT-YOURSELF MICROCOMPUTER RECIPE BOOK that raises the



level of the field handsomely.

While it's stretching things a bit to call this flyer a book, the document is loaded with information the beginner needs desperately to assemble an appropriate system of his own. Not only is there a clean diagram of all the elements that might go into a personal computing system, supported by a conventional component list with prices and a tidy little discussion of what computers are and what they're good for, but a list of sensible system packages is included.

The packages are represented as items on a menu, ranging from The Big Mac through Pete's Whopper and Belgian Jugged Hare to Beef Wellington. The ingredients of these menu items are carefully listed with their prices so that a naive potential computer purchaser can see what different amounts of money will buy for him.

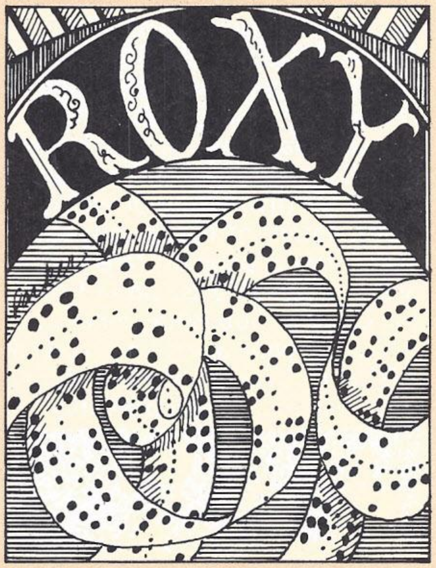
Even if you are planning to buy from a computer store away from the Bay Area and even if you will purchase different brands of computers and peripherals, these packages are extremely useful for guidance and comparison. Of course you will be tempted by Beef Wellington, with its garnish of twin floppies and a Decwriter at $7180, assembled and tested. But your mouth will water, too, at the Captain's Stew with a hearty power supply and a video monitor assembled and tested at $1255.

To take advantage of this free offer, write Computer Center, Inc., 321 Pacific Avenue, San Francisco, CA 94111.

Only a Paper Moon

A Hollywood friend made a cheerful living back around 1960 by producing feature movies for about $20K apiece (no, not porno, adventure) and distributing a few prints of each himself to theaters around the country. "How do you choose theaters and handle bookings?" he was often asked. "Well, I can't afford to send out a print for any booking that doesn't guarantee at least two hundred dollars. I advertise the films in trade papers and any exhibitor who guarantees two hundred dollars plus a percentage of the gross beyond the minimum can have a print for three days."

The pictures weren't exactly blockbusters and the distribution system wasn't highly scientific,



but the man did all right. Times have changed.

20th Century Fox is now using an IBM 370-135 to choose the best and most profitable showplaces for the studio's film releases. A spokesman commented: "Profit from a film can be significantly increased if we place it in theaters with proper revenue histories, and arrange contracts that favor the studio's interests." The computer system provides instant access to information on every American theater that has played a Fox release in the last three years. The system contains general descriptive information on each movie house — location, seating capacity, equipment, parking facilities, etc. . . Also available are statistics on gross revenue by week, advertising allowance, percentage of revenue to Fox and other details of past exhibit contracts. The studio can determine what kinds of pictures do well, where. In fact, the contracts negotiated with individual theaters and the films provided to them vary widely.

"Before we used the IBM system," says the spokesman, "it was just an impossible clerical task to manually examine theater history every time we reviewed a contract. We expect the IBM system to be a major factor in helping us take better control of film placement and, as a result, ensure

us more profits."

Considering inflation, those $20K pictures must now cost $50K and a producer in that league isn't going to buy a big computer to help him sort out bum exhibitors from not-so-bum ones. However, he might swing one of IBM's 5100 personal computers or other microsystems to help him flourish in show biz. The little guy now has big muscle available. Anybody with software and movie-theater data . . . this may be a commercial opportunity for you.

Fortran Already

MICROSOFT, the handful of young guys who developed the highly successful ALTAIR BASIC for microcomputer systems and have been producing proprietary software at a feverish rate ever since, have announced the availability of a FORTRAN IV compiler for the 8080 microcomputer. Called FORTRAN-80, the initial release of this compiler is a full implementation of the ANSI Standard Fortran with the exception of the double precision and complex data types.

FORTRAN IV is the workhorse language of the computer industry and a vast literature of operating programs exists in the language, ready for implementation on personal computers. FORTRAN is regarded as more complex and efficient than BASIC, less easy for the beginner to use, but well within the capability of any serious programmer. Chances are large numbers of packaged FORTRAN programs will be made available, so that computer users can apply their FORTRAN-80 equipped systems to specific tasks without the need to learn the language if they choose not to.

FORTRAN-80 provides three data types including: logical (one byte), integer (two byte), and real (four byte floating point). An extended version of FORTRAN-80 with double precision (see Jef Raskin's *Big Computer, Little Computer* article in the February/March issue of Personal Computing) and complex data types is forthcoming.

The compiler generates pure, relocatable code (may be placed in ROM) and the runtime package may also be placed in ROM. The one-pass compiler requires less than 12K bytes of memory and the runtime system less than 6K bytes.

A relocating linking loader is included with the FORTRAN package. Therefore, subprograms may be compiled separately and linked at load time. This also means that only the specific subprograms required are loaded (including system subprograms).

Another part of the package is a relocating assembler and an assembly language debugging program. The assembler may be used to produce FORTRAN compatible subprograms. The debugging system may be used with the load map produced by the loader to debug FORTRAN and/or assembly language programs.

Additional features of Microsoft FORTRAN-80 include multistatement code optimization, mixed-mode expressions, and all standard FORTRAN library functions for reals and integers.

Individual copies of FORTRAN-80 may be purchased for $500 including documentation. Manual $15. The company is at 819 Two Park Central Tower, Albuquerque, New Mexico 87108.

Welcome Star Trekker

Star Trek mania knows no bounds (see Steve Pollini's Star Trek article in the last issue) and as ST clubs and activities proliferate, it's increasingly difficult to keep track of what's going on. People's Computer Company pointed out in a recent issue that a non-profit service organization called STAR TREK WELCOMMITTEE is available to tell you more than you really need to know about that remarkable show . . . "ST technology, ST actors, details on the making of ST (live action or animation), details of the various episodes, trivia, penpals, fans in your area, revival efforts, aid in forming clubs or publishing zines . . . whatever your question on STAR TREK or STAR TREK fandom . . ." the answer can probably be had by sending a *self addressed stamped envelope* along with the question to STAR TREK WELCOMMITTEE, c/o Shirley Maiewski, 481 Main St., Hatfield, MA 01306.

We haven't yet tested the organization's store of knowledge about the sources and characteristics of STAR TREK computer games, but given the furious activity in the field, STW must surely be lacking some information. Do you have information? Want to be a Good Guy? Contribute your knowledge and some of your time to STW.

And, by the way . . . who is carrying personal computers with video terminals and STAR TREK software to the hospitals where a lot of bright interested people are languishing, wishing for something good to do? Anybody? Want some converts to personal computing? There's a captive audience. Good club activity.

High Level Language for SC/MP

National Semiconductor has developed a version of the Albrecht/Allison TINY BASIC for use with its SC/MP microprocessor system, called NIBL (National is nothing if not cute). Placed in the public domain, the language is available in paper tape form through COMPUTE, the National Semiconductor Microprocessor Users Group newsletter. NIBL will also be available soon as firmware in ROM.

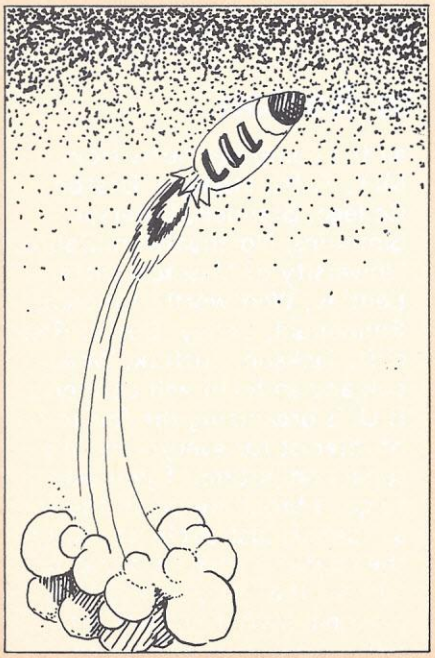
National was acutely aware that the small microprocessors designed chiefly for control systems, rather than general computing, were being used by customers who knew a lot about mechanics, levers, pipes and pneumatics, but almost nothing about electronics and programming. The assembly language of the microprocessors made the devices useless to designers who couldn't take time to learn the secret codes. TINY BASIC/NIBL allows almost anybody to master the microprocessor and design practical control systems.

There's a price, of course. NIBL is slower than assembly language and can't handle operations like video generation or direct control of fast peripherals. On the other hand, for these applications, the programs can be proved out in NIBL and translated into SC/MP machine code by a hired expert for the final, fast system.

NIBL will allow a great many more individuals to get personal with microprocessors.

Basic for the Taking

A BASIC interpreter has been released to the public domain by the Lawrence Livermore Laboratory, better known for its work in nuclear weaponry. The Laboratory operates under the direction of the University of California and the United States Energy Research and Development Administration. This particular interpreter is credited to John Dickenson,

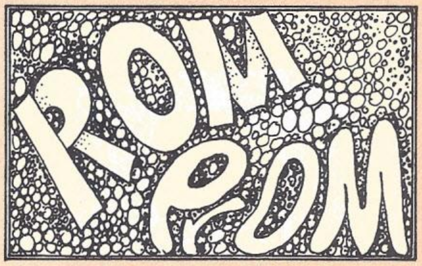


Jerry Barber, John Teeter and Eugene Fisher.

The BASIC interpreter is designed to operate on an 8080 system. It is "pure code" so it can be placed in ROM or PROM. It requires 5K of storage and includes a complete floating point package. This valuable property may develop into an important resource for computer non-professionals.

While LLL is pleased, as well as required, to make its unclassified taxpayer-supported work available freely, the organization is not in a position to debug and discuss the system with outside users. Instead, steps have been taken to offer the material through various distribution outlets whose purpose *is* interaction with the user. One outlet is NTIS, the National Technical Information Service, a government agency. A second is the Intel User's Library. A third outlet, perhaps most accessible to the average personal computer owner is *Dr. Dobb's Journal of Computer Calisthenics and Orthodontia,* Box E, Menlo Park, CA 94025.

Dr. Dobb's Journal is a lively, typographically confusing, mad-



dening, highly entertaining and useful periodical published ten times a year by People's Computer Company. Editor Jim Warren has already run a 46-page listing of LLL BASIC, complete with notes and documentation. Jim shows every sign of sticking with the subject, serving as a referee in its future development.

Dr. Dobb's is well worth its $12 subscription price and reprints of all back issues are available, so newcomers can get in not only on LLL BASIC from the beginning, but the rest of the tomfoolery as well.

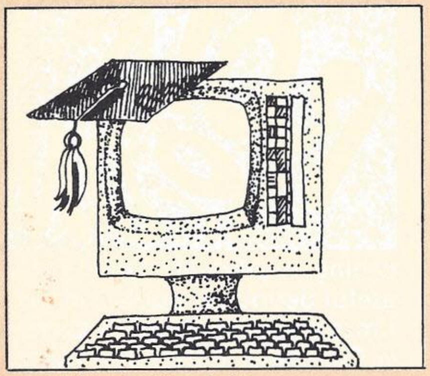
All letters and calls about LLL BASIC are best directed to Dr. Dobb's and the other publishers. There's nobody to complain to about problems with this free material, but a vigorous user's group can make it into another good, standard language.

random access

Become a Computer Expert for $296?

For $296, you can use a microcomputer, PDP-11 minicomputer and PDP-10 maxicomputer including terminals, plotter, line printer, disks, tapes, cards, art and sound studio, and a dozen computer languages; receive six weeks of one-to-one, around-the-clock tutoring; attend introductory classes; participate in a wide variety of advanced seminars; use a library of specially selected texts; and get university credits. It happens this summer at Wesleyan University, in Connecticut.

The course is part of a program that leads to an M.A.L.S. (Master of Arts in Liberal Studies), but you can take the course without



enrolling in the full program, even if you don't have a B.A. The course is open to absolutely **anyone**; it starts at the very beginning and requires no degrees or prior knowledge of math or computers. It requires only 20 hours/week; use the rest of the week to take other courses, relax in the sun, or attend extra computer seminars free. There's enough happening to keep you going 24 hours/day, if you wish.

The basic text is **The Secret Guide to Computers** (see related article), supplemented by several others. The flexible tutoring lets you pursue whatever computer topic interests you most.

The $296 covers the course, registration, and texts. Housing for July 5 - August 19 costs $135, or $200 per couple, or make your own arrangements.

For details, send your name to Russ Walter, 92 St. Botolph St., Boston, Ma. 02116.

Random Access Express

Business Week, Newsweek, the New York Times and New West have all run surprised stories on the outburst of personal computing in this country, but now our field has made the real bigtime with a notice in AMERICAN EXPRESS — a Newsletter for Cardmembers that comes in the envelope with the American Express bill. Among sixteen items in this compact mailer recently was one titled Computers in Your Home.

The writer estimates that thirty thousand or more people now have personal computers of "the digital, problem solving type" and he seems relieved to discover that their uses are becoming "more practical, intellectual." He mentions preparation of tax returns and experiments with computerized music among other practical,

intellectual applications.

While attention from a powerful member of our economic community is welcome, it's a bit disappointing that computers don't sound like more fun in the write-up, more exciting. Is it the nature of computers or the stodgy way they've been presented for decades that prevents feature writers from making them sound genuinely attractive? Interesting, anyway, to see this low-key linking of Amex cards and computers that "can be had for $200 to $1200."

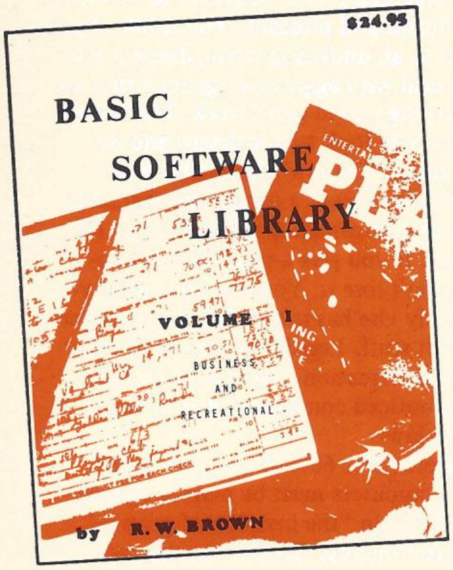
Computer fair comes to Boston

If you are going to be anywhere near Boston on August 4th, 5th or 6th, you may want to get over to Boston University. They are going to have a PERSONAL COMPUTING show complete with exhibits, seminars and everything that's new and up to date that you can see, touch, feel and play with in home computing. And just because Boston suburbia has more colleges and universities per square foot than any other region of the United States doesn't mean that this computing show is just for high school, college students and their professors. To be sure, all the computer buffs from MIT, Tufts, Harvard, Boston College, Brandeis, Emerson, Simmons, Northeastern, Babson, University of Massachusetts, Bentley, Wentworth, Lincoln, Emmanuel, Lesley, Curry, Radcliff, Jackson, Suffolk, Wheelock and so forth will be there. B.U. is organizing the fair to be of interest for everybody in personal computing. If you want to get a brochure on all the exact details just circle no. 100 on the reader service card and we will see that they send you a personal invitation that will tell you what it is all about.

FANTASTIC SOFTWARE

This LIBRARY is a complete do it yourself kit. Knowledge of programming not required. EASY to read and USE.

Almost FREE Less Than $1 / Program Complete



VOLUME I ..$24.95
- BOOKKEEPING
- GAMES
- PICTURES

VOLUME II ..$24.95
- MATH & ENGINEERING
- PLOTTING & STAT
- BASIC STATEMENT DEF

VOLUME III ..$39.95
- ADVANCED BUSINESS
- BILLING
- INVENTORY
- PAYROLL

VOLUME IV ..$9.95
- GENERAL PURPOSE

VOLUME V ..$9.95
- EXPERIMENTER

This library is the most comprehensive work of its kind to date. There are other software books on the market but they are dedicated to computer games. The intention of this work is to allow the average individual the capability to easily perform useful and productive tasks with a computer. All of the programs contained within this Library have been thoroughly tested and executed on several systems. Included with each program is a description of the program, a list of potential users, instructions for execution and possible limitations that may arise when running it on various systems. Listed in the limitation section is the amount of memory that is required to store and execute the program.

Each program's source code is listed in full detail. These source code listings are not reduced in size but are shown full size for increased readability. Almost every program is self instructing and prompts the user with all required running data. Immediately following the source code listing for most of the programs is a sample executed run of the program.

The entire Library is 1100 pages long, chocked full of program source code, instructions, conversions, memory requirements, examples and much more. ALL are written in compatible BASIC executable in 4K MITS, SPHERE, IMS, SWTPC, PDP, etc. BASIC compilers available for 8080 and 6800 under $10 elsewhere.

Available NOW ! ! ! at most computer stores

The "HOW-TO-DO IT" Books for the "DO-IT" Person

Written in compatible BASIC immediately executable in ANY computer with at least 4K, NO other peripherals needed.

Add $1.50 per volume for postage and handling.

SCIENTIFIC RESEARCH

1712-T FARMINGTON COURT
CROFTON MD 21114
Phone Orders call (800) 638-9194
Information and Maryland Residents Call (301)-721-1148

Personal checks — allow 4 weeks for shipping. Pricing subject to change without notice. If air mail shipping is desired add $3 per volume to price, Continental U.S. only. Foreign orders add $8 for each volume.

CIRCLE 8

With Colonel Colt's six-gun, the uneasy average man was equal to the dangers of the frontier. The uneasy average man at today's frontiers of knowledge can hold his own with the personal computer. It is:

THE EQUALIZER

Let us assume that you own a personal computer system. All right, you dirty rat, the government is gonna getcha!

You may think of yourself as an ordinary sort of person, quietly exercising your right to private control of your own thoughts. You hope that your computer system will help you in coping with the complexity of the modern world, in handling your correspondence and taxes, in organizing your activities to be more efficient and pleasant. You think of yourself as an underdog trying desperately to deal with aggressive agencies that are burying you in paperwork. That's not how you look to politicians and do-gooders. The few who realize you exist see you as a dangerous character.

It has been observed that information defies the law of mass/energy conservation. If you give a piece of information to somebody, you don't lose it. You still have what you started with, but somebody else has it *too.* It is not easy for politicians to contain such stuff. Legislative efforts to deal with the conservation of information have not often been effective. You may have noticed continuous warfare over our patent and copyright laws.

Non-technical legislators (over 60% of Congress is lawyers) have decided that computers must be magical, giving their users immoral privileges in "the invasion of privacy" that should be reserved to Congress. To save us, the politicians are actively attempting to destroy the privacy of computer users.

The way legislation is being written, nationally and locally, *any*one who stores and processes information about individuals in an automatic system must inform those individuals . . . by public notice, by letter, whatever . . . that they are listed. You must state the purpose of the listing and the listed individual may have access to all data about himself in your system (presumably at your expense). The information may not be used for any purpose but that declared. Think of it!

Suppose you use your computer system as a journal, making entries about your day-to-day activities, the people you meet, prospective sales prospects, news that may affect you in some way. You can enter information in random order and pull it out in useful order. In theory, you'll have to contact each of the folks to tell them why you are interested in them.

Dear Mr. Mayor,

I have data on your activities listed in my computer system. My intention is to aid your opponent in the next election by telling him in detail what you have been doing as mayor. If you want a listing of your misdeeds (which will cost me several dollars to print out) feel free to demand it.

Yours Truly,
A. Sucker

Dear Mr. Smith,

Your name is on the list of Christmas card correspondents I keep in my personal computer system. I address cards automatically and edit my list yearly. I'm not quite sure who you are, but if you would like to see my speculations and my comments on the doubtful virtue of exchanging cards with you, let me know.

Best Regards,
Your Friend

Dear Mr. Handyman,

You should know that I have been keeping my household accounts in a personal computer system and I have a record of invoices you have given me for services I doubt you actually performed. If you want a peek at my evidence in advance of our pending lawsuit . . .

Dear Mr. Snerd,

I have in my personal computer system a list of prospects for three pieces of land that I own and would like to sell. You are on that list with what information I could find about your business interests. If you would like to strengthen your resistance to my presentation by looking at my information . . .

If laws of this sort were passed with respect to handwritten or printed information, the do-gooders would be outraged and the press would scream bloody murder. However, since not many people yet own personal computers, we have quietly been allowing superstitious politicians to rob you of your freedom.

What about *your* personal right to privacy? Should snoopers rummage at will through your private records merely because you have a personal computer to help you?

Do you believe that the proposed controls will be effective against bad people? If you believe that, you may also believe that nobody in banks is bribed or coerced into giving your records away, that the information you give the census taker in confidence is not used by other agencies, that credit agencies provide accurate information, and that the moon is made of green cheese.

Donn Parker commented during his interview with Personal Computing that some of the computers brought back from Southeast Asia were bugged to broadcast data. People have even been known to copy secret papers illegally. Security in any system is an illusion.

Your local city council may soon demand that computers and programmers be licensed and closely regulated. The effect of laws like this is to beat down the little guy who can't afford the red tape. The government and big institutions will simply grow stronger and more offensive at the little guy's expense.

Luckily, the laws can't possibly work, because the age of the mass-market computer with high power and very low cost is almost upon us. You will soon be joined by many people who share your special problem with personal privacy. In the meantime, you will be hounded and harrassed by every do-gooder who believes that the way to increase his own freedom is to diminish yours. Time is on your side if you aren't beaten in the short haul.

Meanwhile, if you'd like to set a backfire, consider agitating for different legislation. Accept the fact that a significant percentage of data credit agency and government systems is garbled and always will be. It is impossible, with or without computers, to maintain files so accurate that innocent folk are never injured. It's a tough world. The problem is that people *act* as if the information were all accurate. The solution to the problem is not to attempt the impossible, but to dispel the illusion that the content of the files is true.

Let us pass a law that makes it legal, even desirable, to tell lies on a limited scale. Say that one question in ten on credit forms, tax forms, reports to agencies of all kinds, can be falsified deliberately without penalty. Income may be exaggerated. Names may be changed. Ages may be altered. Various colorful lies can be woven to entertain the institutions.

Will this cripple commerce and government?

Certainly not in this day of computers!

At worst, information quality would be degraded by ten percent and probably a lot of people wouldn't take the trouble to lie, so that real degradation would be far less. In practice, the deliberate lies and accidental garbles might not add up to as much as ten percent of the total information stored.

That is, on a statistical basis the information would be ninety percent correct . . . not bad by most standards. The difference would be that every individual case is subject to deep suspicion. No private citizen could be taken for granted. Even the Mafia, blackmailing your bank's loan officer to report your financial status, wouldn't be able to get reliable information.

This is a chance to protect privacy by increasing freedom instead of promoting tyranny that hurts the individual more than it hurts institutions.

Certainly this approach is as practical as legislating against nature, though it may not be as funny.

$194 for the whole kit & kaboodle.

the kit $95

the kaboodle $99

The kaboodle is our popular SC/MP Kit microcomputer package: SC/MP microprocessor, ROM, RAM, timing crystal, PC Board, and all necessary drive logic. All for just $99.

The kit is our new, low-cost terminal kit: keyboard display, and special ROM with debug keyboard/display scanning program. All for just $95.

You don't need a $500 development system.

You don't need a $1,000 teletype.

You buy our kit and kaboodle, and you're in business. In microprocessors, video games, home control systems, whatever. For training, for learning, for development.

If you'd like complete information, we'll send it to you for nothing.

Or, if you'd prefer the actual microcomputer and terminal kit, that'll be just $194 extra.

PC 5/6

National Semiconductor
2900 Semiconductor Drive, Santa Clara, CA 95051
Gentlemen:
__Please send me more information.
__Please send me a real live SC/MP Kit (ISP-8K/200) $99.
__Please send me a real live Keyboard Kit (ISP-8K/400) $95.
Here's my check. (California residents add 6% sales tax.)
Also available at your local distributor.
Name________________ Title__________
Company________________________
Address________________________
City______________ State______ Zip______

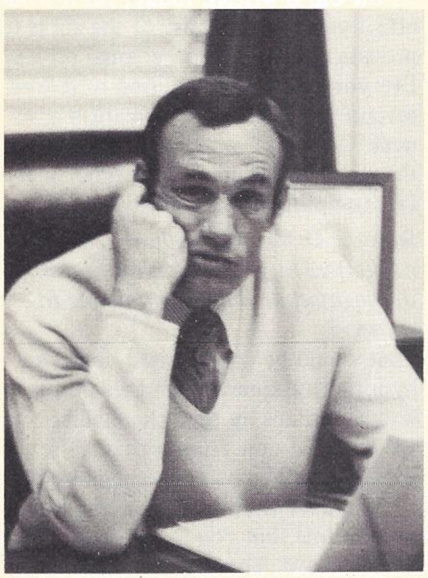
National Semiconductor

See us at NCC Booth 1565

A Personal Computing Interview

Congressman Barry Goldwater, Jr. on

Computers, Privacy and the Law



Congressman Goldwater of California is an active champion of national and international legislation designed to protect the individual's privacy from malicious or indifferent abuse by institutions equipped with computers. He explains what he thinks he's doing in this PERSONAL COMPUTING interview.

Personal Computing: *You've been involved with the issue of personal privacy and computers for some time now. Why?*

Barry Goldwater, Jr.: My concern for individual privacy is a natural extension of my conservative philosophy. It stems from the basic premise that in order for our democracy to operate effectively, individuals have to be free from personal harrassment or intrusion into their privacy. It's a very natural thing for me to flow into.

But there are conservatives in the Congress who aren't heavily involved with personal privacy. How are you different?

The thing that triggered my involvement was my exposure to the FBI's NCIA System. Their capability to store and withdraw information was demonstrated to me on a very routine basis when I went to visit one of their regional offices. My initial reaction wasn't that they were doing anything wrong but that they sure had the capacity to do questionable things from a privacy standpoint. It was from that time on that I began to look into the whole concept of privacy with regard to the responsibilities of the Congress.

You were the sponsor of the Privacy Act of 1974.

Ed Koch, who is a Democrat, and I were co-sponsors of that act on an equal basis. I had introduced my own bill, which was very comprehensive, covering both the government and the private sector.

My approach was to let the courts write the law so it could be enforced by the courts.

Ed wrote a similar piece of legislation, but he wanted to set up a Privacy Commission to do the interpretation.

We both recognized that, in the past, privacy legislation had gotten bogged down in partisan politics, so we married our two bills. We put together our expertise and our own viewpoints and the final result was an equally shared effort by our staffs and ourselves.

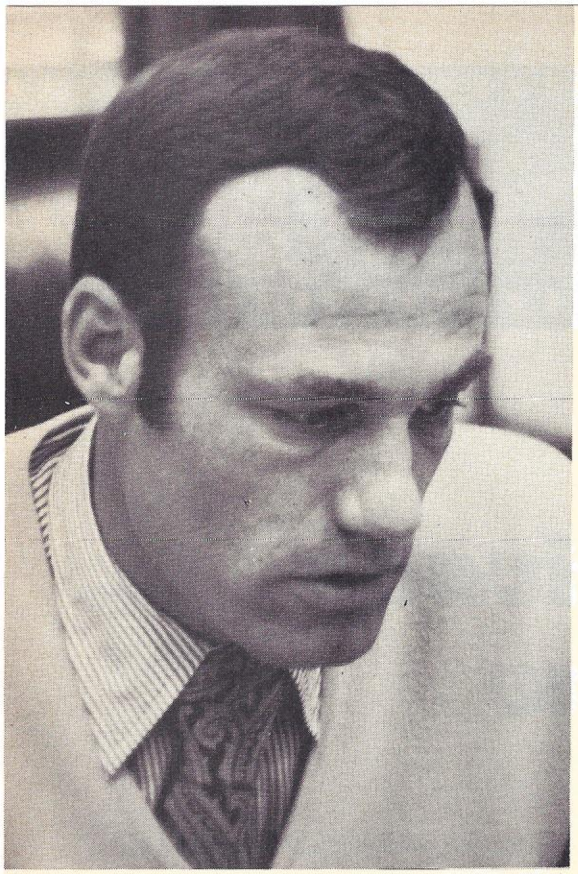
Are you pleased with the results of that legislation?

One of the charges of the Privacy Commission is to examine the privacy act to see if it's doing well or not. From what I know of this, I'm pleased. It's functioning fairly well if for no other reason than it hasn't caused any major catastrophe within the government nor impeded the capability of the government to function.

One of the things I am concerned about is that some agencies may have taken advantage of the flexibility that is in the law. They could declare that a system was normal and routine use and therefore exempt from disclosure requirements. They wouldn't, of course, be exempt from the administrative requirements holding people accountable. I'm not making any accusations, but that's something I want to look into.

Do you get the impression that some agencies are less than enthusiastic about privacy laws?

I don't get that impression. Agencies are not only vehicles to carry out the law



written by the Congress, but they are composed of people and people are resistant to change. I know I am. When you start changing procedures within an agency, you're going to get some resistance, and we expected that.

Looking at the whole federal structure, I have to say that it was received fairly well. As we get a chance to talk with the people in agencies we see a renewed enthusiasm for privacy concerns. I see this with the Department of Defense. Maybe it's just on the surface and I'm being fooled, but I get the impression that they are very serious about privacy. They've testified before our commission a number of times.

What about the Department of Justice?
As it pertains to the privacy act, I really can't comment. As for pending legislation, such as the right to private records bill that Ed Koch and I have introduced, they're not too enthusiastic. It depends on who you're talking to within the Justice Department. I'm convinced that FBI Director Kelly and the FBI are concerned about privacy.

Aren't you concerned that some of your privacy legislation will impede law enforcement?
I'm very strong on law. I support capital punishment, and I think we ought to start putting more criminals away instead of letting them out.

I want us to be tough on crime, but I don't enjoy seeing law enforcement going on fishing expeditions. I don't enjoy seeing the IRS going on fishing expeditions, either. And I don't like to see business abuse individual rights in the name of expediency or economic efficiency.

We've seen an atmosphere evolve in the past few years where an individual, in exchange for certain benefits, is assumed to waive any and all control over information he's required to give up. The right of an individual to his privacy has become subservient to concern for utility, expediency and economic convenience. I think we've got to slow this thing down and get back on the track.

I'm not opposed to the collection of information, because it can obviously be used for the benefit of all. I'm concerned that when information is collected on an individual we allow him an opportunity to make sure the information is accurate, that it's relevant, that he knows what it's going to be used for. In other words, the individual needs to have control.

You now have a bill before the house concerning privacy, HB1984. What will this bill do?
The Privacy Act of 1974 applies only to the federal government. 1984 would take the same kinds of procedures and apply them to the private sector.

The principle is that any system which is secret is wrong.

What are some of these procedures?
For one, you have requirements to make public existence of information systems on individuals. The principle is that any system which is secret is wrong. Individuals controlling these systems would be subject to fines and imprisonment for violations of the law. Right now that doesn't exist.

People would have the right to see all information that is stored about them. They would have the right to challenge and correct that information. There would be restrictions on using information for a purpose other than what it was collected for. Transferring information from one system to another would be restricted. There would be protection for foreign nationals whether they're residing in the United States or not. An accounting of who has access to data files on individuals would have to be maintained.

All these things seem to be written with large data bases in mind. With the advent of personal computers, thousands and perhaps millions of individuals will have their own systems. Will this law apply to me if, say, I maintain a file of my friends or business contacts?
As the bill is written today, yes it would affect you. The purpose of the act is to provide safeguards for personal privacy. Everyone would be required to follow certain principles of information practices. Of course, when the bill goes through the Congress there'll be an effort to define it more clearly.

Suppose there are a million personal computers by 1985. Don't you think this legislation will be impossible to enforce?
That would be complicated, but it's a hypothetical question and we don't know. Perhaps by then there will be a whole set of new criteria that we're going to have to be concerned about.

You have the whole area of computer crime that you interviewed Donn Parker about. This has certain implica-

"... I believe I can tell you that the United States is beginning to desire that discussions begin — and I believe that such discussions will have the objective of leading to bi-lateral and multi-lateral treaties — on international information management ..."

Late in 1976, Congressman Goldwater was invited to Toronto, Canada, to address a meeting on "Trans-National Data Flow." His schedule got out of control and he missed the date, but his good assistant, Joe Overton, stood in to deliver his short, crisp address. PERSONAL COMPUTING offers this excerpt with permission.

We are rapidly approaching an international data-spasm — trans-national data flow that operates under no restraint or accountability. And, if we let this happen, all the various national efforts to balance our rights and need to know with our concern to protect individuality, personality and human integrity will be frustrated or crushed.

What concerns me greatly is the informational pirate. The individual or organization that takes advantage of the international legal vacuum — as did the ocean-going pirates — to pillage businesses and individuals of other nations, take their informational valuables, if you will, and retreat to a safe haven to utilize them for their own gain. Telecommunications is as vulnerable to interception and theft as were ocean-going vessels. And, the international treaties and agreements on copyrights, patents, etc. are not sufficient to afford essential protections and provide appropriate redress for injury. If the United States is any example, our national courts still have no clear-cut legal concept of informational privacy, they have no consistent precepts of the relationship between the need to know and the right to be left alone. However, between many of the nations of our contemporary world, gentlemen's agreements and "understandings" can and do exist so that situations can be worked out on a case-by-case basis.

What none of us can protect against is the pirate and the potential data-haven that he can retreat to.

It is the possibility of data havens that concerns me greatly. Much of what the various nations active in information management law have to date created in the way of national statutes can be married in bi-lateral or multi-lateral agreements. What we all are ignoring is the individual or nation that for other than political, ideological or sovereign reasons may pirate information. In America, Watergate, organized crime and international corporate bribery have given us a glimpse of what the potential is. Each of us knows that telecommunications satellites and microwave communications can be intercepted, signals separated and deciphered. Each of us knows that these communications devices can be used to carry or transmit non-contracting, third party communications. Each of us knows that mass production and increasing availability of equipment have brought each of us to the brink of individual international communication capability. All that is so far thankfully missing — and I could well be misinformed on this — is a deliberate, organized effort to do it as a business.

All of the incentives are there. Previously, information of only national significance now has international value — and it can be acquired and utilized almost instantly. International marketing techniques have been turned into a science, with many talented practitioners needing work. Technicians have almost become citizens of the world, having a greater allegiance to their art and craft than to anything else. Finance is available; so is the supplier of the hardware. And, there is now a well defined international appetite for information — and an equally well developed digestive system to put it to profitable use.

Thus, I would hope that as we as nations are concerned, aware individuals grapple with the problems of how we make compatible freedom of information and personal privacy, how we humanize information practice, and how we intelligently marry our various national efforts with those of other nations — and as I am sure you perceive, I fervently hope we do just that — we will also consider how we can prevent and cope with the data haven potential. I consider it a probability that data havens will develop; the world has renegades and pirates in every human activity. To fail to come to grips with this problem may well mean that technology, at least informational technology, will master the master.

The time to start is now, and I for one am looking for allies.

tions as to what kind of regulations we're talking about, what kind of protection we're going to be able to afford.

Do you think it will become necessary for the government to license people who use computers?

No, I don't think that is going to be necessary. As a matter of fact, I'm hopeful that 1984 as it is currently written does not have to become law. The point I make is that individuals involved in record keeping have a responsibility to apply common sense principles to their work. When I speak to people about privacy legislation I tell them they can help us lessen the reason for 1984. And this applies to General Motors as well as to you as an individual.

You would prefer that the government not have to legislate privacy legislation.

Yes, it's obvious to me that when you pass a law to solve a problem you inherently create another problem. Any time that you begin to restrict the natural flow of human beings in their day-to-day life, you're creating a lot of problems that you're going to have to tackle later.

Oftentimes I'm reluctant to see new laws being enacted. I wouldn't at this point, with the knowledge that I have see any reason to license computers or even computer programmers.

What are some of your feelings about computer programmers?

I would like to see computer programmers elevated to a level of real importance within a corporation. That way they could take on a real professional stature, which would call for greater responsibility on their part.

I would like to see the establishment of associations within business and industry designed to ensure individual rights and the safety and protection of information systems. In my way of thinking, it's good business to be concerned about individual rights and privacy. And I think it's less expensive if they do it themselves than if it's imposed upon them by the government.

Realistically, though, you can talk about voluntary compliance, but what are the chances of this happening?

I think they're fairly good. I'm optimistic and I'm encouraged by what I see. Some of our larger corporations are moving in this direction. They're analyzing their information systems and revamping them.

These corporations are leaders and they're very visible. Smaller businesses can see what they're doing and hopefully they will follow suit. That's one of the things I'm looking for, and I'm

sure the Privacy Commission will look into that too.

You get into problems when you talk about how much business can afford to spend to apply the principles.

In addition to 1984, you have another privacy bill before the house, HB1985, which concerns itself with banking.
The reason for 1985 stems from the recent Supreme Court decisions of Miller vs. the United States and from California Bankers Association vs. Shultz. These two decisions clearly said that your bank records don't belong to you, they belong to the bank. And this could be extended to any kind of personal records including telephone records and credit records.

This is offensive to me and to Ed Koch and just about anyone else we talk to . . . clearly offensive. Even though the law has never spelled it out, there's a sacred trust that you find between a lawyer and his client, a doctor and his patient or a bank and its customers. That's one of the reasons our banking system works. People trust the banks. These decisions clearly destroy that perception.

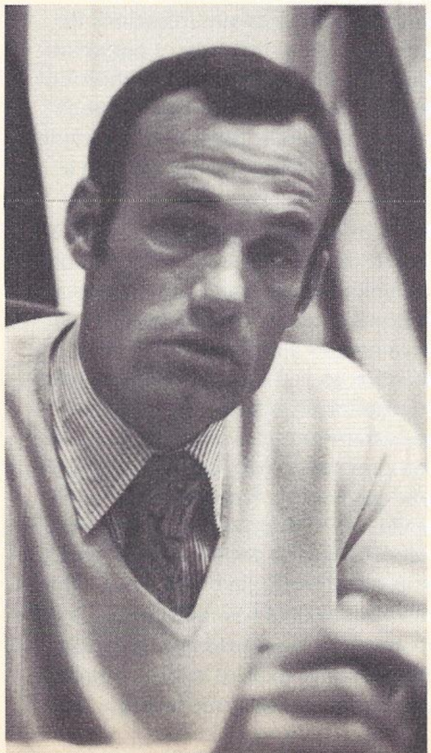
So you're disappointed with the Supreme Court when it comes to the rights of individual privacy.
I've been somewhat suprised at the current Supreme Court. In fact, I find myself more personally aligned with the late Justice Douglas who said back in 1967 that privacy involves the *choice* of an individual to disclose or reveal what he thinks he possesses. I have no argument with that. On the other hand, I find that Justice Rehnquist is more conservative on this issue than I can accept, and he is clearly becoming a spokesman in the Supreme Court on privacy. I find myself at odds with his interpretations.

Don't you, as a conservative concerned with privacy, find yourself at odds with many groups who you would have normally allied with?
Well, I would say that oftentimes people interpret my actions as being anti-business, or anti-law enforcement. I think this is a wrong interpretation. I was a businessman for nine years prior to coming to Congress. I will not take a back seat to anyone when it comes to defending free enterprise and the right of anyone to go into business and earn a decent living. I've told you my feelings about law enforcement.

To what extent does privacy legislation become a partisan issue?
I cannot really say that it's a partisan issue. Anytime you talk about privacy to someone who's not thought much about it, they don't disagree with you.

Where you run into problems is in the application of privacy in the real world. We ran into this in the course of the privacy act of 1974; where certain agencies of the government were able to develop allies in the Congress and as a result, we had to compromise. Civil Service was one example. Law enforcement, FBI and CIA functions – we had to compromise. It wasn't political as much as it was objections by certain agencies that were able to make their case to a sufficient number of members in the Congress.

At PERSONAL COMPUTING *we hope the proliferation of computers will actually lead to an increase in personal freedom and power. The automobile has certainly increased our personal freedom.*



I don't think I would say that. Any technology is a two-edge sword. Sure the automobile has allowed the individual to get out of his home and be more mobile, but it's also polluted the air. I don't think that is being free – having to breathe dirty air.

Airplanes have the capacity of increasing our standard of living by improving commerce, and we've all benefited from them. But the side effects are noise and pollution. You always have the other side of technology.

Computers are obviously going to enhance commerce, they're going to increase the individual's ability to compete to improve his situation, but you always have this potential of privacy abuse. It can get out of hand and you'll have no control.

How about the people who control computer technology?
Today, I think they are concerned about privacy abuse. Yesterday they weren't.

I want to be tough on crime, but I don't enjoy seeing law enforcement going on fishing expeditions. . . and I don't like to see business abuse individual rights in the name of expediency or economic efficiency.

The technical community oftentimes does not look on the political or social side of things. They look purely at the nuts and bolts of a technical problem. But we're finding that that is changing. I suspect we're evolving a whole new generation of scientists and technologists who are looking at the broader aspects of their work.

In the computer community we're seeing people trying to develop hardware-secure if not software-secure systems. That's encouraging.

When you first became involved with the privacy issue, did you realize what you were getting into?
We had no idea that the privacy issue would blossom to the degree that it has. My staff and myself spend an enormous amount of time on this. Joe Overton, my legislative assistant, winds up going all over the country speaking about privacy. I get invited to speak. We're even beginning to get involved with some of the international aspects of privacy. It's very far reaching.

The new generation of Diskette Drives is here and under control.

PerSci has it–a family of diskette drives "design-years" ahead of competitive drives–now available in complete low cost subsystems for interface to 8080, 6800 and other major microprocessors.

The Highest Performance Diskette Drives:
PerSci diskette drives, both single and dual head units, offer a combination of performance features unique in the marketplace while still maintaining compatibility in existing systems:

☐ Voice coil positioning for access speeds seven times faster than competitive drives (76 tracks in 100 ms)
☐ A low power all DC system reduces cost and assures high reliability
☐ Automatic electric loading simplifies operation and protects media
☐ Small size permits 5 single drives or 4 dual drives to be mounted vertically in a 19 in. rack

The Most Powerful Diskette Drive Controller
The PerSci Model 1070 Diskette Drive Controller puts the advanced performance of PerSci drives to work in microprocessor based systems. An IBM format compatible, "intelligent" controller, the Model 1070 will handle from 1-4 drives with minimum demand on the host system.

In fact, with addition of a power supply and keyboard to the PerSci subsystem, the user can perform many floppy disk routines without additional hardware or software.

Controller features include:
☐ 8080 microprocessor
☐ Formatter/control IC
☐ 4K disk operating system
☐ 1K data buffer
☐ 8080 or 6800 microprocessor interfaces
☐ RS-232 serial interface optional
☐ IBM data format

An Economical Diskette Drive Subsystem
A complete subsystem including a single diskette drive (Model 70), the Model 1070 controller with interface and a controller-to-disk-drive cable is available in single units or OEM quantities. For double capacity, a dual diskette drive (Model 270) subsystem is also available.

Don't settle for yesterday's diskette drive. Get the new generation under control from PerSci, 4087 Glencoe Avenue, Marina Del Rey, CA 90291 (213) 822-7545.

PERSCI

Peripherals a Generation Ahead.

CIRCLE 9

Shopping for a computer at the ByteShop is almost as much fun as building one.

Computers are fun. And affordable. Thousands of people are already using personal computers for TV games, video color graphics, digital music and lots of things nobody ever dreamed of —till now.

Until we came along the toughest part about getting started with computers was shopping for one. Now you can visit a Byte Shop and put your hands on a wide variety of personal, hobby and business computers. Browse through our library of helpful literature. Ask a few questions. Or sign up for one of our starter classes. We'll take the time to advise you on how to get started— or even if you should get started at all.

Computers don't make the least expensive hobby in the world —just one of the most satisfying. You can get started for $35—the cost of our starter class. If you're ready to start building your own computer from a kit, that will cost you about $250. Or, we'll sell you an assembled computer for $695.

To get you going quickly, we'll let you buy now and wait 90 days before your first payment.

What are you waiting for? Come on in.

Arizona
Phoenix—East
813 N. Scottsdale Rd.
Phoenix—West
12654 N. 28th Drive
Tucson (not open)
2612 E. Broadway

California
Berkeley
1514 University Ave.
Burbank
1812 W. Burbank Blvd.
Campbell
2626 Union Ave.
Diablo Valley
2989 N. Main St.
Fresno
3139 E. McKinley Ave.
Hayward
1122 "B" St.
Lawndale
16508 Hawthorne Blvd.
Long Beach
5453 E. Stearns St.
Mountain View
1063 W. El Camino Real
Palo Alto
2233 El Camino Real
Pasadena
496 S. Lake Ave.
Placentia
123 E. Yorba Linda
Sacramento
6041 Greenback Lane
San Diego
8250 Vickers-H
San Fernando Valley
18424 Ventura Blvd.
San Francisco
321 Pacific Ave.
San Mateo
1200 W. Hillsdale Blvd.
Santa Barbara
4 West Mission
Thousand Oaks
2707 Thousand Oaks Blvd.
Westminster
14300 Beach Blvd.

Colorado
Arapahoe County
3464 S. Acoma St.
Boulder
2040 30th St.

Florida
Cocoa Beach
1325 N. Atlantic Ave., Suite 4
Miami
7825 Bird Road

Indiana
Indianapolis—North
5947 E. 82nd St.

Minnesota
Eagan
1434 Yankee Doodle Rd.

New York
Levittown
2721 Hempstead Turnpike

Oregon
Beaverton
3482 SW Cedar Hills Blvd.
Portland
2033 SW 4th

Pennsylvania
Bryn Mawr
1045 W. Lancaster Ave.

South Carolina
Columbia
2018 Green St.

Utah
Salt Lake City
261 S. State St.

Canada
Vancouver
2151 Burrard St.

Japan
Tokyo
Towa Bldg., 1-5-9
Sotokanda

BYTE SHOP™
the affordable computer store

CIRCLE 10

Wisdom from the professionals

Some historians maintain that when the written language of a society contains more than 64 characters, the general membership of that society is not literate. There are exceptions to some degree, but human beings traditionally have not expected all their children to learn disciplines above a certain level of complication.

In societies like those of ancient Egypt or even modern China, literacy is a *profession.* You go to a professional scribe to get letters written just as you go to a butcher to have animals slaughtered and subdivided for consumption. Hieroglyphics and ideograms are just too clumsy for general use. Reports from China today indicate that most of that country's scientific literature is published in English and other non-Chinese languages. While massive efforts to promote general literacy are being pressed in China, the language itself is an overwhelming obstacle to the cause, and numbers of scholars and political observers are pessimistic about general Chinese literacy *in Chinese.*.

Computer languages, all artificial, are to the general population in the western countries as Chinese is to the general population in China. Professionals and dedicated amateurs may learn the art of their use, along with the skills to manipulate the equipment, but there is no immediate likelihood of general computer language/system literacy.

The revolution is under way, and it is at least plausible that computers will be used as commonly as pencils after another decade or two. But it's clear that the complexity of the systems must be reduced (or human nature altered) before that can happen. Nobody knows at this point how the changes are to be accomplished. Control of computer development is still almost entirely in the hands of professionals who have disciplined themselves to do the necessary work. Professional experience is still our only practical guide to what may be done with computers, and non-professionals may gain useful insights from the trade-talk of the pro's. Personal computer users will find their own concerns strangely echoed in the experience of others. Some examples:

An Interface Business Series Editorial

International Computer Programs Inc. of Indianapolis publishes magazines dealing chiefly with proprietary computer software – packaged programs for business. This abbreviated commentary from the Winter 1976 issue of *Interface* will sound familiar to many.

"Incredible. Don't believe it. History is mirroring itself: like Narcissus, in love with its own reflection.

"You would have thought we'd have learned by now. The past years have been a learning experience for people in data processing. Judging by what we absorbed we must have been nodding off in the back row. Remember 10 years ago: proprietary software wasn't exactly a buzzword. More like a buzz-off. It was new, untried, untrusted. Thwarted at every turn by NIH (not-invented-here). Software product was an outsider trying to smuggle itself into a bundled world, trying to spark some excitement. It caught on like a wet match.

"Suddenly the earth shook, the sky fell, the waters parted, and Mother unbundled. Gradually, people became aware of the packaged alternative to skyrocketing programming costs: software product. Off-the-shelf, over-the-counter, under-the-table, behind-the-back; whatever, it was a time and money saver. Programming solutions could be bought for 1/10 the cost to develop them; they could be up and running in 1/40 the time.

"Then along came the minis. Naturally, we apply all that newly acquired wisdom about alternatives to minicomputers, right? Wrong.

"Look around you: We're starting from scratch again. Mini software is out there, but downwind of us: the world hasn't sensed its presence yet. You wonder why? Because there's not a valid reason in the world for it.

"At least part of the blame rests with the DP industry's trade publications. They used to snub minis; today they pack off their entire editorial staffs to mini country and devote complete issues to them. They're not dumb. The trade pubs see what is prophesied to be a $30 billion industry by Orwell's target date, and they know there's a byte of the pie for them.

"The problem is: there's something missing from these incisive articles. When these editorial staffs return to pen their wisdom, people are awed. By hardware stories. Then they start talking software, and it's 1966 revisited. The only difference is that now they're writing about little computers instead of big ones. That aside, the software sections are as informative as a newspaper morgue.

"What irks is the mention of all the other aspects of expense. Like all the hidden ones. And the chief hidden one, the alltime thorn-in-the-side, is the expense of programming. In article after article, people achingly lament the escalating costs of developing software for their minicomputers.

"Prospective buyers are instructed to consider the 'extra' costs of the mini buying. Hardware cost; hardware maintainence; programming costs; program maintainence; cost of computer operation; cost of facilities; cost of conversion; cost of on-going management support. Important items, all.

"Yet in each of these state-of-the-articles you see these factors mentioned only in reference to (a) in-house programming, and (b) hardware manufacturers' software. In spite of all we've learned about proprietary software product, it doesn't even get last page ink.

"It should at least share second-billing with in-housers and vendors. Yet there are problems even with that. Let's face it: hardware vendors are in the business to sell machinery. A few of them won't even touch applications software.

"Re programming in-house: you already know the story there. In most cases, an in-house developed program will be better than a purchased product; it can be sculpted to perfection by Cobol-minded Michaelangelos. The problem is economic: three, maybe four per cent of software development efforts pan out within the original budget. Or the original time frame.

"The way to beat this problem lies frequently with proprietary software: the (thus-far) ignored alternative. For one thing, it takes the 'guess' and the 'work' out of guesswork. When a prospective mini buyer (of which there are many in the insurance industry) sits down to tabulate all of those 'total cost' factors, software product gives him some concrete figures with which to work. As with any software buying, it's necessary to pick the vendor judiciously. Find one with a product that meets a high percentage of user requirements (usually 80% to 90%) and who includes installation (conversion), training enhancement and support in one fixed price.

"The result is, the buyer has fewer variables in his 'total data processing cost' equation. At least three of those factors can be lumped into one fixed price. The buyer can much more viably operate within his budget. He can be up and running in a fraction of the time. And his programmers won't waste their time on banal redundancies. Instead they can refine the purchased software: they can create.

"And there are software products out there for minis. The suitcase computer industry is young, but precocious. And a great many products are for insurance applications.

"Insurance companies are going to minis inside their home offices as well as outside them. And the consensus is apocalyptic: users say minis are the best things that ever happened to them. Some companies are even investigating the feasibility of *replacing* the big mainframes with a series of minicomputers. Could be you'll take a museum tour in 1990 and see a 370 setting between a White House tape deck and a Brontosaurus rib: extinct things.

"Regardless, the market for minis is big and getting bigger. And the problem remains that all those computers will have to be programmed. When you're contemplating software alternatives for yours, keep in mind the proprietary one. It's unsung, but only for now."

John Peers on the Wing

John Peers has been described as "the Monty Python of the Computer Industry." He's an entertaining, fast-talking Englishman with an exciting approach to the problems of making computers useful to the rank beginner. The system produced by his company (Logical Machine Corp., Sunnyvale, Calif.) is hardly for private owners, with its $40,000 price tag. However, the approach embodied in the Adam system may be the wave of the future for non-professionals.

PERSONAL COMPUTING hoped to persuade Peers to write a crisp, coherent article on the philosophy he and his associate, Gerard Horgan, have developed toward the use of computers in business, but the man can't be boxed that way. Instead, we offer a bouquet of interesting passages, some slightly repetitious, all enlightening, taken from writings and the talks Peers often delivers to business gatherings. These selections come from: The Trouble with Computers. . . is People, The Domesticated Computer, Business Computing or You Don't Need to Buy the Whole Laundry to Wash a Pair of Socks, and How Come the Faster and More expensive the Machine the Less It Does for Me?

The Domesticated Computer

Whether we like it or not, computers are going to become a part of our lives rather more personally than they have been up to now. Specifically, I mean that within the next 10 years we will enjoy a computer in almost every home as a standard item. . . just like a refrigerator, dishwasher or garbage disposal . . . .

You are probably saying, if that is the case, why aren't we all using them in the home right now? One reason is that at this time they're too expensive to buy. Secondly, ordinary people can't program or teach the computer what to do. The third and most important reason is that today people can't talk to a computer with their voice . . . . Everyone must wait until the computer is domesticated.

It won't be long before the computer can actually recognize words in simple sentences. From then on it's only a matter of building a syntax before the computer can understand reasonably colloquial language.

Beyond the Hobbyist

Personal Computing should become *the* slogan of the new breed of computer users. Unfortunately, unless we are all very careful, it could become associated – exclusively – with the computer hobbyist approach. I am not putting down the computer hobbyist, you understand; it's just that I don't want to see this aspect of one of man's most exciting challenges become burdened with a "make-do" approach. You know, rather like a poorly produced fiberglass body sitting on top of a rather old and mangled VW chassis, the whole thing pretending to be a Ferrari. No, I define personal computing to be the desire for people to get *what* they want *when* they want it instead of being told what they can have when somebody is ready to give it to them. . . and usually at exorbitant cost.

You have heard it before, but there *is* a revolution going on in the computer business, whether we like it or not. I happen to like it. When a revolution like this is happening, the best thing to do is understand it, take part in it and get maximum benefits. Doing the ostrich bit is rather a waste of time.

The computer business has been around for 30 years and we have come a long way since those two monsters on the East Coast in 1946. Now we are at the *beginning* of the micro-computer chip era. O.K., so we have come a long way, but why has so little filtered out as benefits? I suspect that nobody has reassessed how we cause the things called computers to function. The problem lies in what I call human feedback. Let's start at the beginning.

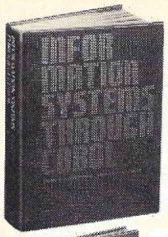
Scientist comes along and invents a computer, then says: "What shall I do with it?" He answers: "I shall deal with a scientific problem." He defines what the problem is, invents the method for the solution, implements the method

Introductory offer to new members of the COMPUTER PROFESSIONALS' BOOK CLUB

any *two* of these great professional books for only $2^{95}

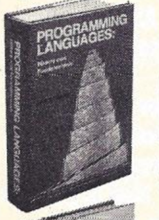
values up to $58.50

Two special bonus books come to you for $2.95 with your first club selection



497/672
INFORMATION SYSTEMS THROUGH COBOL
by A. S. Philippakis and L. I. Kazmier
Pub. price, $15.95
Club price, $12.50

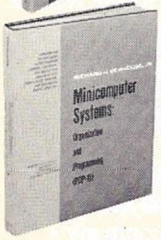
768/03X
MICROPROGRAMMABLE COMPUTER ARCHITECTURES
by A. B. Salisbury
Pub. price, $13.50
Club price, $11.40



767/513
PROGRAMMING LANGUAGES
History and Fundamentals
by J. E. Sammett
Pub. price, $21.00
Club price, $17.50

767/815
GAMES AND PROGRAMS
Mathematics for Modeling
by R. R. Singleton and W. Tyndall
Pub. price, $13.00
Club price, $10.75

767/661
INTERACTIVE COMPUTER GRAPHICS
by B. S. Walker, G. R. Grund and E. A. Drawneck
Pub. price, $14.50
Club price, $12.30



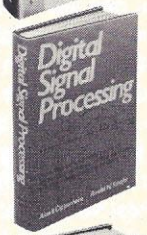
768/641
MINICOMPUTER SYSTEMS
Organization and Programming (PDP-11)
by R. H. Eckhouse
Pub. price, $15.95
Club price, $12.50

767/262
ENGINEERING CYBERNETICS
by R. M. Glorioso
Pub. price, $16.95
Club price, $13.95

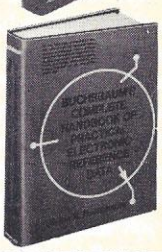
767/98X
THE PROGRAMMERS INTRODUCTION TO SNOBOL
by D. Maurer
Pub. price, $13.50
Club price, $11.40

769/29X
THE INFORMATION SYSTEMS HANDBOOK
by F. W. McFarlan and R. L. Nolan
Pub. price, $30.00
Club price, $22.50

435/278
MICROPROCESSOR APPLICATIONS MANUAL
by Motorola, Inc.
Pub. price, $28.50
Club price, $22.50



767/009
DIGITAL SIGNAL PROCESSING
by A. V. Oppenheim and R. W. Schafer
Pub. price, $22.95
Club price, $18.95



770/271
BUCHSBAUM'S COMPLETE HANDBOOK OF PRACTICAL ELECTRONIC REFERENCE DATA
by W. H. Buchsbaum
Pub. price, $17.95
Club price, $13.50

Be sure to consider these books as well—

766/770
MINICOMPUTER SYSTEMS
Structure, Implementation, and Applications
by C. Weitzman
Pub. price, $19.50
Club price, $14.50

768/714
APPLYING DATA STRUCTURES
by T. G. Lewis and M. Z. Smith
Pub. price, $15.95
Club price, $12.25

574/456
COMPUTER ALGORITHMS & FLOWCHARTING
by G. A. & J. B. Silver
Pub. price, $8.55
Club price, $7.25

767/53X
PROGRAM STYLE, DESIGN, EFFICIENCY, DEBUGGING AND TESTING
by D. V. Tassel
Pub. price, $12.00
Club price, $9.95

save time and money
by joining McGraw-Hill's new

COMPUTER PROFESSIONALS' BOOK CLUB

THIS new professional club is designed to meet your day-to-day on-the-job needs by providing practical books in your field on a regular basis at *below publisher prices.* If you're missing out on important technical literature—if today's high cost of reading curbs the growth of your library—here's the solution to your problem.

The Computer Professionals' Book Club was organized for you, to provide an economical reading program that cannot fail to be of value. Administered by the McGraw-Hill Book Company, all books are chosen by qualified editors and consultants. Their understanding of the standards and values of the literature in your field guarantees the appropriateness of the selections.

How the Club operates: Every month you receive free of charge *The Computer Professionals' Book Club Bulletin.* This announces and describes the Club's featured book of the month as well as alternate selections available at special members' prices. If you want to examine the Club's feature of the month, you do nothing. If you prefer one of the alternate selections—or if you want no book at all—you notify the Club by returning the card enclosed with each *Bulletin.*

As a Club Member, you agree only to the purchase of four books (including your first selection) over a two-year period. Considering the many books published annually, there will surely be at least four you would want to own anyway. By joining the club, you save both money and the trouble of searching for the best books.

MAIL THIS COUPON TODAY

COMPUTER PROFESSIONALS/Book Club
P.O. Box 582 Princeton Road, Hightstown, New Jersey 08520

Please enroll me as a member and send me the three books indicated. I am to receive the two bonus books at the introductory price of $2.95 plus my first selection, plus tax, postage, and handling. If not completely satisfied, I may return the books within 10 days and request that my membership be cancelled. If I keep the books, I agree to take a minimum of three additional books during the next two years at special Club prices (guaranteed 15% discount, often more). I will receive the Club Bulletin 12 times a year. If I want to examine the featured selection, I need take no action. It will be shipped automatically. If, however, I want an alternate selection—or no book at all—I simply notify the Club by returning the convenient card always enclosed. I will always have a minimum of 10 days in which to return the card and you will credit my account fully, including postage, if this is not the case. Membership in the club is continuous but cancellable by me at any time after the four-book purchase requirement has been filled. *This order subject to acceptance by McGraw-Hill. Orders from outside the continental U.S. must be prepaid. Company, business, or institutional tax exemption status is not applicable to purchases made through individual Club memberships. All prices subject to change without notice. Offer good for new members only.*

Write Code # of 1st *bonus* book selection here	Write Code # of 2nd *bonus* book selection here	Write Code # of 1st book selection here

Name______________________________

Address______________________________

City____________________ State__________ Zip__________

EXTRA SAVINGS: Remit in full with your order, plus any local and state tax, and McGraw-Hill will pay all postage and handling charges. **P39215**

CIRCLE 11

in programming and (lo and behold!) now has a computer that actually does something. However, there's an absurdity built into that.

In the *real* world, nobody knows what he wants to do in detail *until he begins doing it.* When you start doing something, you learn more about what it is you wanted to do, then promptly change your mind about what you thought the problem was, hence what the solution should be. I coined Peers Law to describe this about three years ago: "The solution to the problem changes the nature of the problem." This is the basic problem in computers that I'm trying to point out.

The bigger and more expensive the computer monster is, the more likely it is to be hidebound by a priesthood of people enslaved to its every whim. They must work further and further in advance of your definite requirements in order to get the monster programmed and utilized efficiently. That's fun as far as their professional attitude is concerned but bears no relationship to the needs of the real person who actually wants to have the thing do some work, today, NOW, *his way.*

This is why the majority of *detailed* systems proposals are a waste of time and money. Statistics on the installation of business mini-computers support this view. *Eighty per cent of all software which is satisfactory at the time of installation gets changed within two years. The cost of software is twice the cost of hardware over the first two-year period of installation.* Frightening statistics. How do we get around this inconsistency and make the things workable?

In my opinion, one answer is to get rid of the concept of programming as a step which is inherently different from the application of the solution. One step is to accept the idea that the need to make the computer efficient is hopelessly out-of-date. Somebody said to me recently that technology has given us computing power to burn. It is true. Why worry about the most efficient utilization of the processor or the quickest version of a program? As far as I can see, the main consideration is how to get the job up and running, find out what's wrong with it ('cause you *will* change your mind), put that step right quickly and *get on with the job.* When the system works to your satisfaction and you have the time, *then* clean it up.

How do we do all this? There are many approaches, but one thing is for sure: you'll *never* do it if you have to learn a "language," which takes you at least two years if you have the special kind of skill it takes and want to become really efficient. Somehow we must build the skill into the machine. After all, that's what we invented it for.

A Self-Programming Computer

If you look at the desirable qualities or characteristics of computer programming, they are roughly as follows:

1. Entirely logical	3. Absolutely accurate
2. Very repetitive	4. Exceptionally boring.

And if you examine the jobs a computer is good at, they are:

1. Entirely logical	3. Absolutely accurate
2. Very repetitive	4. Exceptionally boring.

From this data, the balance of the equation is obvious. Business needs a computer that can program itself. Business people do not want to understand how computers work; they just want to use them. We don't need to know how a gearbox works to drive a car or how a TV tube works to enjoy Kojak. Can we have a self-programming computer?

Yes.

The Computer as Electronic Pencil

I should admit my biases. My company makes a computer called Adam that is relatively language independent. We are proud of Adam, because a naive user of the system can make it perform useful, complex tasks by giving it instructions in plain language – teaching Adam to understand special terms, rather than learning a special Adam-language. Note: I am not saying that anyone, including us, has found the complete answer. I'm saying that if the situation in general is rotten today, it's going to be absurd in a few years, because you are going to be able to buy complete mini/micros for $100 or less and complete business systems in the $3,000 to $4,000 range. Unless we change our methods, you're going to be faced with a programming bill of perhaps 10 to 50 times the cost of hardware. That's almost like General Motors selling you a Chevette for $5,000, then insisting it is too complicated for you to drive so you have to hire a chauffeur at $18,000 a year.

We *must* design the machines so that they are language independent, so they work in ordinary conversation, so we get away from keyboards and on to voice response. We must make hardware systems so fast internally that it doesn't matter how sloppy we are at getting them to do what we want them to do, as long as they do it, *now.*

Business people are already able enough to run their business with a manual system. Why must we insist that it's more complicated to do it on a computer? It isn't. The computer should be transformed into what amounts to an electronic pencil. The user should be able to pick it up, figuratively, and *use* it. Then we get real computer utilization.

Changing the Nature of Nature

It is exactly the problems which exist prior to the ownership of a computer which are analyzed and programmed and put on the computer. I think you will agree that if the existing problems of a company change upon acquisition of the computer, it is little less than foolishness to go to the trouble of coding a complete, integrated system (which was defined and thought to be required) prior to the acquisition of the computer. In fact, there is some evidence to show that the whole tone of a company changes after it has decided to acquire a computer.

The classic problem is that the data processing staff is usually far more interested in the *computer* than in what the computer is being asked to do, solving the company's problems. It is fun to write real-time executives and compilers. It's really not quite so absorbing to make sure that figures reconcile and balance. It is very boring to have to go back and change a program just because somebody in sales happens not to like it and will not use the perfectly accurate information coming out of the current computer program.

There never seems to be an easy way to make changes quickly and one can never stimulate the computer department to realize the actual needs of the company as a trading entity, serving the purpose for which the computer was purchased in the first place. In a conventional situation, the problems would appear to be built in by the very nature of the tool.

Consider the people who are thinking about getting a computer but haven't yet. Are they frightened of it? Of course they are. The word computer has overtones of fright and horror, coupled with extraordinary high cost and usually crippling inflexibility.

I'll come back to where I started: It is not the computer that can't accept change; it's the humans who have not yet found a way of getting the change onto the machine quickly.

CIRCLE 12 ▶

Introducing the Micropolis MetaFloppy.™ The first system that actually combines the high-capacity storage of an 8″ floppy with the low cost and convenience of a 5¼″ format.

Metafloppy goes far beyond any ordinary floppy. Giving you the lowest obtainable cost per thousand bytes of floppy disk storage. From anyone. Or anywhere.

In fact, there's really no comparison. An ordinary 5¼″ floppy provides just 70K (or so) bytes of storage. Not nearly enough for meaningful work.

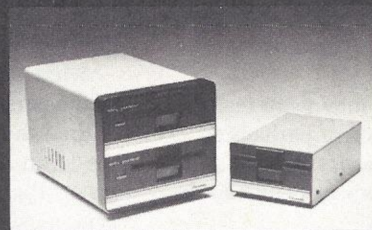
Metafloppy, instead, packs each disk with a whopping 315K bytes of storage! The true equivalent of any typical 8″ floppy. And over four times the capacity of a typical 5¼″ floppy. (We call it "quad" density.)

Best of all, Micropolis delivers everything you need — the drive, S-100 controller, interface cable, power supply, even extended BASIC software — all for an unbeatable $1095.

Or save $150 and buy a Micropolis "double" density drive instead. You get the same complete package but with 143K bytes of storage per disk. That's still twice the capacity of a typical 5¼″ floppy. And twice the buy.

For maximum convenience — and savings — both densities are also available in dual drive models.

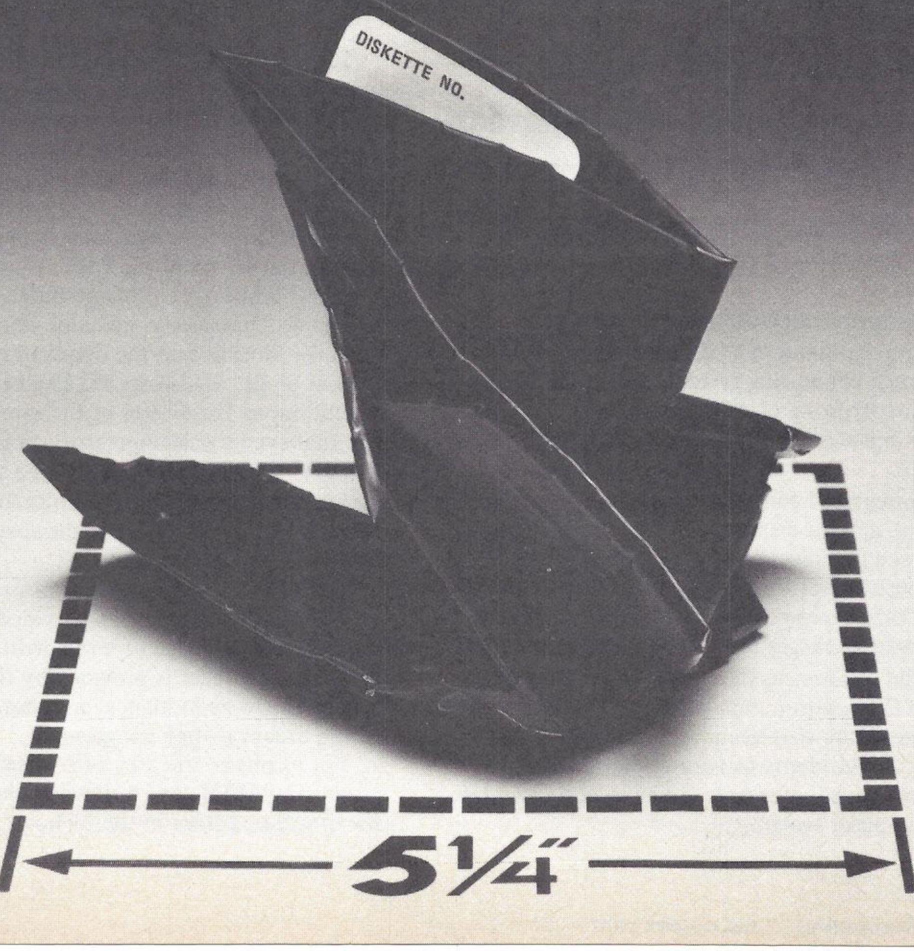
But then, at Micropolis our goal is to give you twice the bytes per buck. And today, it's quite literally the very least we can do.



MICROPOLIS™

In the U.S., please contact:
Micropolis Corporation, 9017 Reseda Blvd.
Northridge, California 91324 (213) 349-2328

It's one thing to get an 8″ floppy into a 5¼″ format. It's another thing to do it without spending a wad.



An analogy may be helpful here. In the days of the stage coach, somebody came along and developed the concept of the railroad train, and it was a tremendous improvement. Man being the creature he is, however, he soon accepts a new tool as "normal" and begins looking around for something better. Trains were wonderful, but came with their own problems. Before you could run them, you needed tracks and before you could lay tracks, you had to buy the land. Before you could buy land, you had to go out and find where the tracks had to go. And before you could do that properly, you had to know where the major cities were likely to grow.

Then, having built the network of stations, tracks and schedules, you found that the resultant structure was so horrendously complicated that it couldn't easily be changed to meet the changing needs of the people using it. In fact, you couldn't go *where* you wanted to go, *when* you wanted to go, *how* you wanted to go. Out of the problem, which had been originally a superior solution, developed the motor car that overcame these limitations.

The difference between cars and trains may seem really slight, but the difference resulted in the sale of perhaps a million times more cars than trains. The car is regarded as normal now, and we think only of its drawbacks while looking for a better solution.

The computer, a superior tool now accepted as normal, must evolve into a more useful dynamic form. It must be very easily changed by anybody who's had a little bit of training equivalent to a course in driving a car.

We have to get rid of programming. Yes, that's true, we have to get rid of programming. The concept of a language that is structured, of compiling, editing, assembling and all the rest of the paraphernalia that goes with programming, is as limiting to the computer as the track is to a train.

Of course, there will still be room for the large programming jobs in exactly the same way as the freight train is an essential part of our transportation system. But would-be computer users cannot be constrained by having to explain to a trained programmer what their problems are. They must not be constrained to using a good solution to the problem they unfortunately defined wrong. Importantly, as the price of human labor goes up, the use of inexpensive hardware mustn't be limited by the need for expensive, unique, highly trained, highly specialized human beings.

The word computer has too many hard overtones, mathematical overtones, high cost overtones. I feel we need a word that derives from an alternative use, perhaps something to do with the concept of a Business Logical Machine . . . a BLM. *(Or a Personal Logic Machine . . . a PLM?: Ed.)* Consider the new generation of machines as BLM's.

What the BLM must be able to do is deal in the language of the person using it. That doesn't just mean that it works in English for English people, Arabic for Arabs, Spanish for the Latin American world (which it should do also). It means that the words which have a unique meaning to the person using the BLM must mean precisely the same thing to the BLM when they are using it. We must have a machine where the choice of "what words mean" is up to the user. That sort of thing is what we've been doing. We haven't solved the *whole* problem. There is increasing awareness now, that we have made the first approach to the general solution. We'll have to wait for the future to see what happens.

Is Adam the first of a new breed?

The dreams John Peers weaves in his entertaining chatter about putting computers under personal control of their owners may be coming true.

With its price tag of forty-thousand dollars (or a mere twenty-thousand in a smaller version) ADAM is not yet a personal computing system, except for the rich. That's *today*! The offspring of ADAM may populate the world tomorrow and some preparation for that experience may serve us well.

ADAM is an integrated computer system including a computer, a keyboard, a video display, a printer, and a rigid disk memory, all housed in a nice, plain L-shaped office desk. Nothing is obviously novel about this. Many companies sell small businesses hardware packages of this general sort, with compatible software packages for special jobs like accounting.

And there's the difference. When you buy ADAM from Logical Machine Corporation, you don't have to buy a package of complex software that becomes your master. At least, that's the plan. ADAM adapts to *your* operations instead of forcing you to change all your procedures to accommodate the inflexible computer system.

To begin with, ADAM is *fast*, with 200 nanosecond cycle time. ADAM also has a 32K RAM, using 16-bit words.

This combination of speed and capacity allows ADAM to control printer, disk and display functions directly from RAM, reducing the need for peripheral controllers that consume both time and money.

ADAM asks questions, quietly, in lower-case type that appear on the screen of the CRT. When the operator answers, giving ADAM orders, the operator's entries appear in upper-case type. This clearly indicates who is boss, or at least provides the illusion that the operator is in full control.

Best of all, ADAM speaks English (or French or Swahili, depending on the desires of the user) in a sort of pidgin dialect that even a naive operator can figure out and respond to. Further, ADAM doesn't give you a standard program to use for each task you might like to perform. Instead, ADAM gently coaxes from the user the specifications of the job to be done . . . and does that.

The operator gives orders in a straightforward vocabulary of English nouns and verbs provided to him. Aha! The user must learn a language of terms with very precise meanings? Well, yes. His relief is provided by the system's tolerance for awkwardness, redundance, and imprecision in the sequence of the orders as they are given.

For example, the user with a need to handle payroll accounts in ADAM can sit down at the machine with a simple list of existing steps in the by-hand procedure already being

used and teach ADAM that procedure, step by step. For example: To calculate Gross Pay, ADAM might be taught –

MULTIPLY	REGULAR HOURS by HOURLY RATE
MOVE	PROD to REGULAR PAY
COMMENT	PROD must be moved before it is destroyed
COMMENT	by the next multiplication
MULTIPLY	OVERTIME HOURS by OVERTIME RATE
MOVE	PROD to OVERTIME PAY
MULTIPLY	DOUBLE TIME HOURS by DOUBLE TIME RATE
MOVE	PROD to DOUBLE TIME PAY
COMMENT	This last time may be unnecessary if double
COMMENT	time pay is not required later.
ADD	PROD to OVERTIME PAY
ADD	SUM to REGULAR PAY
MOVE	SUM to GROSS PAY

That's not very complicated and when the procedure expands to handle withholding taxes, insurance fees, bonuses, and all that, it simply grows longer, not really more complicated. Almost anybody can talk the machine into doing what he wants. He can teach ADAM new words and new tricks.

ADAM can remember lots of information, not only new instructions and routines, but data. That rigid disk represents a memory that is, by personal computing standards, colossal. It handles 5.3 million bytes. Indeed, the capacity is twice that, but everything is recorded redundantly for protection. The data transfer rate is so fast and memory search is so fast that the user at the keyboard can't tell that he isn't using the whole system as RAM. You can stuff data and instructions into ADAM for a long time before that memory is filled.

Does memory actually get filled? "Yes," says Gerard Horgan, the system's technical developer. Do you add memory? "Not usually," says Gerard in a manner that makes you realize you've asked something silly again. "In most cases, we chat with ADAM's owner about what he really wants to keep in memory. People often save things they don't need and some of them can be disposed of, leaving useful capacity. Futhermore, most of the instructional procedures stored in ADAM's memory are written by users who don't care about efficiency. We can copy all of the material, carry it back to our own people, and rework the material so that it takes half of the memory. The user can't tell what we've done, because ADAM's performance with the reworked material is just the same. ADAM asks all of the right questions and gives all of the right answers. The user knows only that he has plenty of memory again."

That's the great secret of ADAM. The system can be operated extremely inefficiently, from the point of view of computer purists, but to the complete satisfaction of users who don't care how clumsy the machine is, so long as they can use it easily.

Maybe ADAM is a representative of the future, when tiny packages will contain computing power and capacity far beyond the needs of the average user, and at cost anybody can afford. Peers admits that he now oversells ADAM "just a bit." While it is true that even a naive subject can sit at ADAM's keyboard and make the system perform usefully under Gerard's frustrated tutelage, that same naive subject can't remember what he learned in an hour and keep ADAM going usefully.

Learning to handle ADAM is like learning to drive a car. At first, your whole attention is devoted to getting a car started, pressing the accelerator the right amount, shifting, steer-

This is ADAM, not the lady looking doubtfully at the printer, but the computer system, including the desk. Actually she's seated on top of the disc memory, which is housed with the very compact computer in the corner section of the desk. The lid lifts off the desk to give access to the memory when disc-packs are to be changed, and when the computer or memory need service, the whole system can be pulled up to desk level, where the technician can reach everything in comfort.

ing, and not killing the engine, all at once. After a few hours, the novice driver acquires the necessary reflexes to keep the machine in operation and can begin to think about navigating through traffic. Same with ADAM. The novice needs some practice so he can concentrate on the job he's doing instead of the procedures for making ADAM work. Still, that's greatly different from spending months or years in learning the theory and practice of programming and *then* having to learn how to run a machine.

When ADAM first appeared in 1975, Bob Patterson of Minicomputer News offered this cautious commentary that still applies.

Is ADAM for real?

ADAM, the "no-software" business computer, exists. The problem in writing about it is that its developer, John Peers, does not want it described or thought about in computer terms. His reason for taking this approach is reasonable: he does not want to frighten first-time users who, understandably, are wary of computers.

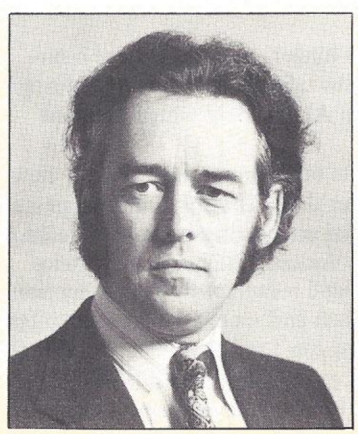
Also understandably, some of Peers' statements therefore unnerve computer people, who want to relate ADAM to what they know about computers. We can report that during about an hour's hands-on experience with ADAM, experience that deviated from the "canned" demonstration, we discovered that ADAM, like any computer, is really not the least bit forgiving. But the machine's prompting cleverly conceals the fact that the programmer is being forced to observe a normal series of logic and format restraints.

We think programmers will find ADAM fascinating. Only time will tell how first-time users get along with ADAM.

Will ADAM's philosophy enter personal computing?

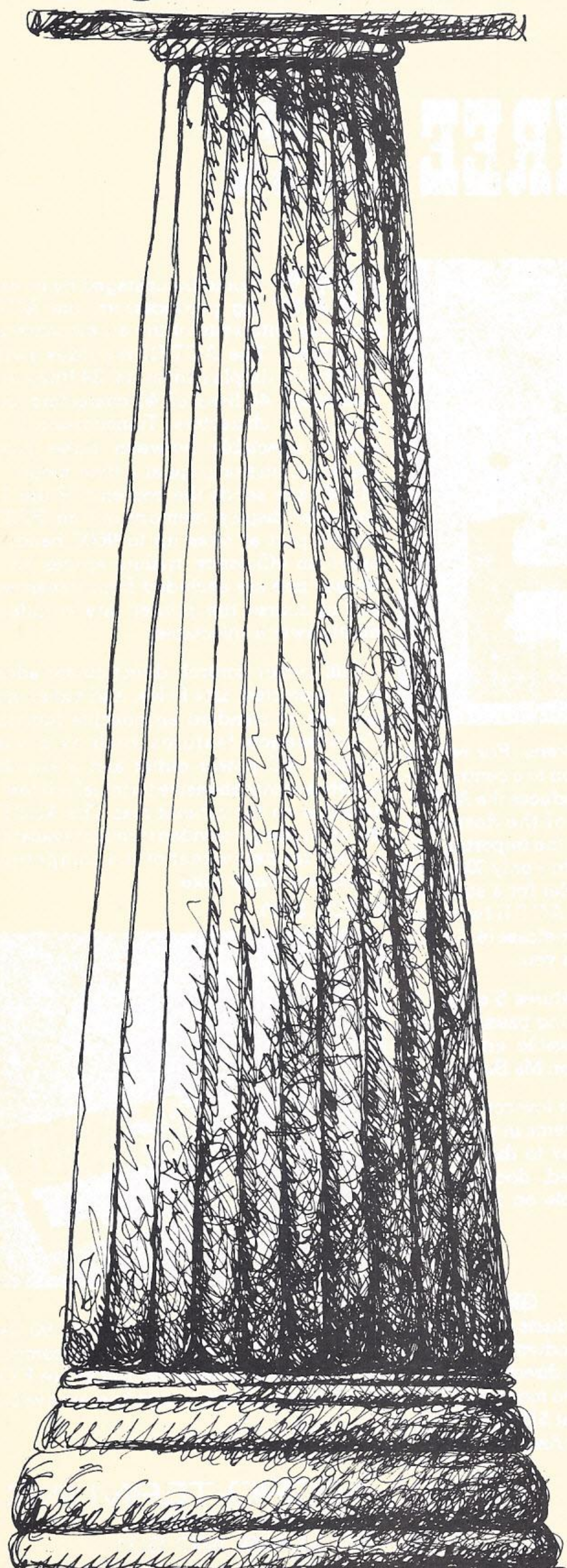
Time will tell.

John Peers is president of Logical Machine Corporation, ADAM's maker, so to speak. Says Peers, "I am allowed to say all the entertaining and upsetting things about computers and even to suggest technical approaches that might be practical for converting my ideas to reality, but of course the really difficult task falls to Gerard Horgan, who really makes these things work."



Gerard Horgan directs research and development for Logical Machine Corporation and is the technical brain behind ADAM. Now that ADAM is performing handsomely in the marketplace, Gerard has grown a distinguished-looking beard. That beard, the classy British accent, and the general air of disapproval he carries with him (does it seem to you that he enjoyed having this picture taken?) add a lot of color, that's colour, to his demonstrations of the system.

SOFTWARE COLUMN



by Bill Gates
President, Microsoft, Inc.

The languages used on microcomputers are becoming as well developed and sophisticated as those used on large computers. APL, COBOL, FORTRAN IV and RPG II are all available now or under development. Microcomputer software is still very primitive in the area of operating systems, yet the functions of operating systems, which I will describe, make them particularly useful for microcomputers.

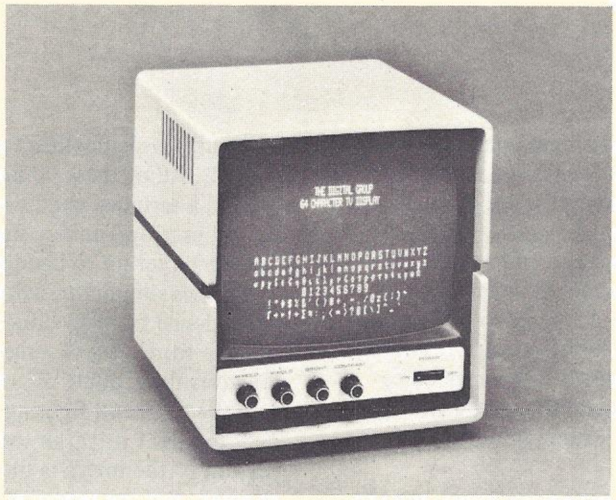
At present personal computer users can be grouped into two classes. One class of users buys all their equipment from one company and uses the software that the manufacturer provides without modification. This type of user can get by without knowing about IN's, OUT's, status bits and interrupts but is severely constrained in expanding his or her system, or using it in any way the manufacturer hasn't provided for. The second class of user is willing to get his or her hands dirty and decode and patch software that is provided. This user must learn machine language and spend a lot of time getting the various components in his or her system working together. The key to getting the best situation for all users, allowing them to customize their system without learning the complexities of I/O (input/output) and investing a large effort, is to provide personal computers with a good operating system.

An operating system is such an important element in large computer systems that when programmers refer to the computer they usually mean the computer along with its operating system. Only the system programmer who modifies the OS (operating system) itself has to worry what channels the I/O devices are on, what special protocols have to be used to communicate with them and how file data is structured on a disk or magnetic tape. Other programmers can act as though opening a disk file or sending a line of characters to the line printer are built-in operations, just like the machine instruction that adds two registers. To initiate an OS operation, a single call to the OS is inserted in a program at the appropriate place. When this call is executed the OS takes over and goes through whatever complex procedure is necessary. The programmer who used the call may never have heard of parity, longitudinal checksums, or cyclical redundancy codes (all three of which are used to detect device errors).

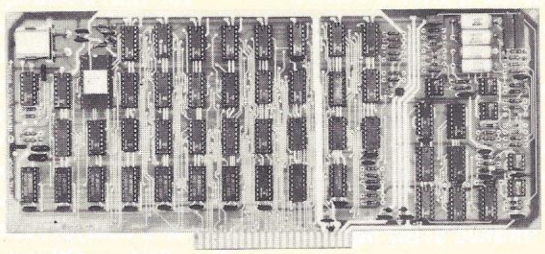
The functions of an operating system can be broken into four areas: 1) I/O handlers, 2) memory management, 3) commonly used commands and 4) common routines. By far the most important is area one, the I/O handlers. These routines make it unnecessary for any program running under the OS to perform I/O operations. On 8080 based microcomputers this means no IN's or OUT's need to be included in a program. On 6800-based microcomputers this means data never has to be transferred into or out of the I/O registers. To do input or output you first call the OS with the name of the I/O device you want and a number indicating the type of I/O you want to do. This is the OPEN call. Examples of device names are TTY (for teletype), LPT (for line printer) and CDR (for card reader).

The two most common I/O types are binary and character. In binary input mode the OS receives a stream of 8-bit

The Digital Group adds character(s).



64, to be exact.



The Digital Group's computer systems have a lot of character already. Just one quick look at any of our products in their unique custom cabinets confirms that. But we believe it never hurts to add a bit more.

So, the Digital Group has added character in a big way to give an added dimension to the operation of our video-based computer systems. We are pleased to announce our new TV readout with a 64-character line. It will give your system a great deal more capability. Give it more character, if you will.

Here are the specifics on the Digital Group TV Readout and Audio Cassette Interface:

1024 Character TV Readout

- 64 characters horizontal by 16 lines
- 7x9 character matrix (effectively 7x12 due to character shifting)
- 1K on-board RAM for buffer storage—requires no main memory—completely independent
- 128 character ASCII
  - Upper case alpha
  - Lower case alpha with base line extenders (g, j, p, y)
  - Numbers and extended math symbols
  - Greek alphabet
- Software driven cursor—forward and backward
- Compatible with most microprocessors; Interfaces with 1 8-bit parallel output port
- Timebase may be driven with an external timebase (may be synchronized to TV camera, TV set, etc.)
- Readout timebase available at connector (can be used for graphic driver, etc.)
- White characters on black, and/or black on white; software selectable
- Plugs into standard dual 22-pin TVC connector on Digital Group Systems

Improved Audio Cassette Interface:

- Reliable FSK recording technique
- Uses standard unmodified audio cassette recorder
- Write cassette system uses a digitally synthesized frequency shift system, derived from TV system's master crystal oscillator
- Read cassette system easily aligned using the write system as an alignment aid.
- Runs at 1100 baud (100 characters/second)—loads 16K in 3 minutes

512 TVC to 1024 TVC Upgrade Kit:

As always, when the Digital Group extends the capabilities of our systems, it doesn't mean obsolescence for any products. We are offering an upgrade kit for present Digital Group system owners who wish to go to the longer line length. This kit uses most of the IC's from our TVC-F readout. No unsoldering is required; all new sockets, capacitors, resistors, PC board and other necessary parts are supplied.

Prices:

TVC-64—Full 64-character TV Readout & Audio Cassette Interface:

Kit — $140 Assembled — $205

TVC-64UPG—Upgrade kit from TVC-F:

Kit — $65

If you already own a Digital Group system, our 64-character line will definitely enhance its operation. If you're just looking, you might want to keep in mind that the Digital Group has a lot of characters.

Write or call now for details on our new 64-character TV readout and all our other exciting products.

the digital group INC.

box 6528 denver, colorado 80206 (303) 777-7133

CIRCLE 15

SOFTWARE COLUMN

characters from the device and passes them unaltered to the program requesting input. Binary mode is used for saving and loading memory images. In character input mode an entire line of characters terminated by carriage-return is received at once. The characters are echoed and special characters are used for editing the input line. Some character, usually rubout, causes the previous character to be deleted. A line-feed character will be output after carriage-return is echoed so that only one character needs to be entered to end a line. Character I/O mode is used for sending and receiving lines of text, as in an editor when you want to receive a command line.

After a call to the OS to open a device for input, the OS will return to the calling program with an indication that either there is no such device, or that the I/O mode selected is illegal for that device or that the device is open and you can proceed to do input. Each open device is given a channel number and in calls subsequent to the OPEN call, the device being referred to is indicated by passing the channel number as an argument of the call. After a successful open, the operating system is called every time a program wants to receive a character or group of characters. Each input operation will return with an indication that there was an error, that no more characters are available, or that the characters requested were read in properly. When you are finished doing input, a CLOSE operation should be performed so appropriate action such as turning the device off, allowing it to be used for input on another channel, or rewinding it can be done by the OS. Output is identical to INPUT except that the character stream is passed from the program to the OS instead of from the OS to the program.

Of the many advantages gained by doing I/O through an OS, the most obvious is simplicity. The programmer doesn't have to bother with the setup of a device, the shutdown procedure, making sure that it has input data available or that it is ready to receive output. For CRT's, scrolling, clear page, hold-and-continue output, and cursor positioning require special code. For some terminal devices tabs have to be translated into spaces. For others there has to be a delay between sending a carriage return and the following character since it takes more than one character time for the print head to return to the left-most position. Errors, paging, character translation and buffering also require special attention. Usually a character like break, control-C, or escape is used to signal an alert condition which must be detected even if the program which opened the device is not requesting input. The alert character is commonly used to stop an operation in progress. In BASIC, the alert character, control-C, stops the program that is running. Also, the column that the terminal print head is at must be kept track of and an extra carriage return generated when a line is completely filled. Some devices send a bit at a time, others 7 or 8 bits at a time, and some send a whole line at once. Errors often cause special interrupts.

Timing requirements sometimes dictate that a real-time clock be used and tasks stored in a queue to be activated after a specific period of time. An example of this is a communications channel that sends a bit of data every 3 milliseconds. All of these complications demonstrate that I/O even to a terminal can be extremely complex, and that it simplifies programming significantly if every program doesn't have to be set up to handle these things.

Besides simplicity, working through an OS provides independence from specific device characteristics. If a switch is made from a small screen CRT to a large screen CRT and OS saves the trouble of going back to every program that uses the CRT and figuring out how to modify it. Instead, only the CRT driver in the OS is modified. (Technically, resolution of the image is the same whether the CRT is large or small, but the practical truth is that people can read information on a big CRT more easily. The number of characters per line and number of lines displayed is commonly altered to suit CRT size.)

The final advantage of using the I/O handlers of an OS is flexibility. Say you have written a program to summarize data stored on cassette. Originally you might want your output to go to your teletype so you can tell your program is working. Then you might want to print the entire summary on the line printer. Finally you might want to archive a copy on a cassette and transmit it to another computer. Without an OS, your otherwise simple program must be able to do I/O to a teletype, line printer, cassette and communications line. With an OS you can have your summary program input from the teletype the name of the device to do output to. By typing "TTY", "LPT", "CAS" (for cassette) or "COM" (for your communications line), you can change the device the summary program OPENS and output will be directed appropriately. No part of your summary program will be specific to the device chosen for output. Even if you hadn't decided which line printer to buy when you wrote the summary program, as soon as you interface your line printer and add the line printer driver to your OS, you can use the summary program with the line printer simply by entering "LPT".

I will just briefly mention the other functions of an OS since I am mainly concerned with the advantages of device-independent I/O. Memory management involves allowing portions of memory to be saved in binary, printed out (often called dumping) or loaded-in in binary. At command level, that is when the OS is accepting commands from the console terminal, the verbs "SAVE", "DUMP" and "LOAD" are usually used for these functions. These functions can also be invoked by a program call. If a compiler uses so much memory that it has to be broken up into different phases, a LOAD call can be used by the first phase to bring in the program for the second phase.

Commonly used commands include verbs to examine and modify memory, test system components and print out time as calculated by the real-time clock handler. These commands are included in OS's as a convenience because they are frequently useful.

Common routines come as a side benefit of the other functions in an OS. These other functions require subroutines that are of a general nature, like printing a number in

GET THE BEST FROM... SEALS

Let me get the best . . . Please send the following

- ☐ 8 KSC 500ns — ☐ KIT $295.00 ☐ ASSMB'LD $349.00
- ☐ WWC — ☐ KIT $37.50 ☐ ASSMB'LD $47.50
- ☐ 8 KSC-Z 250ns — ☐ KIT $295.00 ☐ ASSMB'LD $349.00
- ☐ BBUC — ☐ KIT $55.00 ☐ ASSMB'LD $68.00
- ☐ EXT Extender card $29.00
- ☐ 100 pin edge conn — ☐ ALTAIR® $9.00 ☐ IMSAI® $9.00
- ☐ ASSEMBLY & OPERATING MANUAL $4.00

NAME ______________________
PLEASE PRINT OR TYPE
ADDRESS ______________________
CITY ____________ STATE ________ ZIP ________

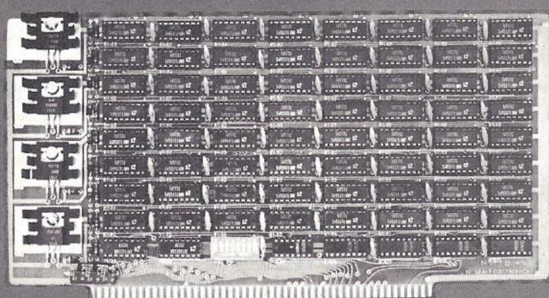
SEND CHECK • MONEY ORDER • COD'S ACCEPTED • CREDIT CARDS

SEALS ELECTRONICS, INC
P.O. BOX 11651
KNOXVILLE, TN 37919

MOST ORDERS SHIPPED WITHIN 10 WORKING DAYS

BANKAMERICARD — COPY ACCOUNT NUMBER FROM YOUR BANK AMERICARD — MY CARD EXPIRES

master charge — COPY ACCOUNT NUMBER FROM YOUR MASTER CHARGE — Copy number above your name on Mastercharge — MY CARD EXPIRES



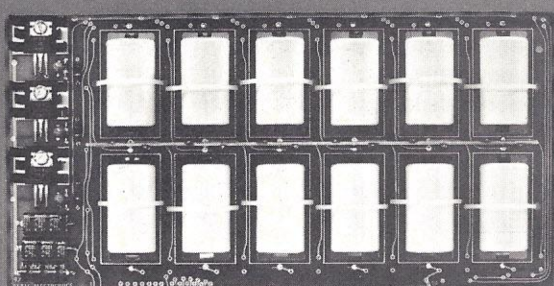
8 KSC [8K STATIC MEMORY BOARD]

Our most popular item. Hundreds of satisfied customers. We have received an enormous number of letters praising our 8 KSC board. Our 8 KSC is undoubtedly the highest quality and most dependable board on the market today.

INTRODUCTORY OFFER

We are proud to announce that you can order your 8 KSC-Z with 250ns memory at the same-yes SAME-price as our 500ns memory.

		KIT	ASSEMBLED
8 KSC	500ns	$295.00	$349.00
8 KSC-Z	250ns	$295.00	$349.00
EXT extender card		$ 29.00	$ 38.00
100 pin edge conn. [Altair®]		$9.00	
100 pin edge conn. [IMSAI®]		$9.00	
Assembly & Operating Manual		$4.00	



BATTERIES NOT INCLUDED

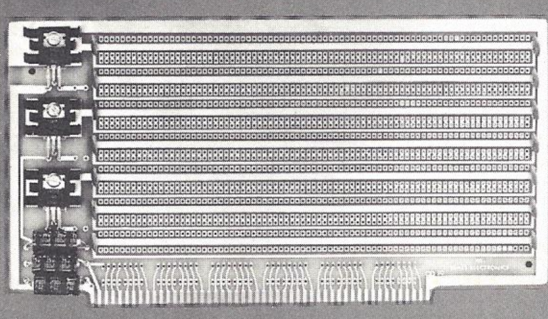
BBUC [BATTERY BACK UP BOARD]

- Automatic battery charging circuit
- Selectable standby voltage outputs
- Will hold up to 12 "C" cell Ni-cad batteries. As much as 12 Amper hrs
- The BBUC comes selected for 2.5 volts standby to pin #14 on the S-100 buss structure, to power up the 8 KSC memory
- Can be wired to back up any memory card which has battery standby capability. Even **TWO** polarities at one time
- Eliminate cluge wires on top of memory
- **Heavy G-10 glass epoxy PC board**
- **Heavy plated through holes-.5 mil. tin minimum**
- **Solder mask both sides**
- **Component layout screened on component side of PC board**

KIT - $55.00 ASSEMBLED - $68.00 ASSEMBLY & OPERATING MANUAL - $ 4.00

WWC [WIRE WRAP CARD]

- Accepts **ALL** IC wire wrap sockets 40, 22, 16, 14, etc.
- 3 voltage regulators: +12v, -12v, +5v
- 3 separate input capacitors 100 ufd
- 14 .1 ufd decoupling capacitors



- Gold plated edge contacts
- **Heavy G-10 glass epoxy PC board**
- **Heavy plated through holes -.5 mil. tin minimum**
- **Component layout screened on component side of PC board**

KIT	ASSEMBLED
$37.50	$47.50

SEALS® ELECTRONICS, INC.

TELEPHONE # 615/693-8655

CIRCLE 16

SOFTWARE COLUMN

hex, printing a string, doing address arithmetic or inputting a number in hex. The calling address and conventions of these subroutines can be provided to programmers using an OS and instead of rewriting these subroutines and putting them in every program that needs them a programmer can simply use calls to the appropriate subrutines in the OS. A standalone program, that is one that runs without an operating system, must have provisions for every device it is going to work with. At the time ALTAIR BASIC was written no operating systems, good or bad, existed, so it had to be stand alone. To support the five protocols used on MITS I/O boards, a complicated scheme involving reading the terminal device type from the front panel switches was implemented. ALTAIR BASIC is now able to support a cassette, line printer, and floppy disk in addition to a terminal, but only those that do MITS type I/O. Homebrew or non-MITS peripherals must either be identical to the MITS counterpart or special patches must be made to BASIC. MITS provides the locations of the I/O routines that must be changed, but it is still difficult to get BASIC working with a special device. Fortunately, the flexibility ALTAIR BASIC provides by allowing BASIC programs to do INs and OUTs makes the patching procedure unnecessary for slow devices. A BASIC program can do I/O directly by executing IN and OUT which are provided as statements in BASIC without BASIC itself knowing about the device being used for I/O. The problems caused by BASIC's being stand-alone have made it clear that some kind of OS is necessary for microcomputers, even if they don't have a disk.

Currently most personal computers are provided without an operating system or with an extremely limited one. The software sophistication of the manufacturers realize the importance of software, but much too slowly to keep up with user needs. Because of this shortcoming, these manufacturers have not realized the importance of providing a good operating system. Writing such as OS to support the variety of personal computers now in use would be a very difficult job. How do you accommodate a user with 4K memory and a tv terminal as well as a user with 64K memory a line printer, a teletype and two digital cassettes? What formats do you support in areas which there are no standards? How can the OS be flexible enough to handle the myriad of devices currently available, not to mention those that will become available in the future? How do you let a user customize the OS for his needs? These problems can be solved properly, but not without a large expenditure of professional effort. Some individual users have developed OS's of their own, and in some cases they are quite well written. However, most of these are specific to the systems they are being used on. Even if they were general purpose though, every user having his own operating system largely defeats the purpose of having operating systems in the first place. If Brand X Company sells a thousand digital cassettes, each purchaser of this device will have to write a driver to include in his particular OS, so a thousand different drivers will be written. This is wasted work. Only one driver should have to be written and Brand X Company should be the one to do it. Also, if you have your own personal OS, every time you receive a program from a fellow user or software house written to run under a different OS you have to modify its OS calls to work with your OS. Any program you don't have the source for, or at least the positions of its OS calls, can't be set up for your system. Even with a source the modification of the OS calls is often a large task. Another problem of having lots of OS's is that the commands, alert character and editing characters will be different for each one. This means time wasted writing manuals and reading manuals and typing "rubout" when you mean "back-arrow" or vice-versa.

The best thing for users would have been if all the manufacturers of personal computer hardware had got together years ago and decided on a standard OS. Every time a new device was introduced the driver needed in the standard OS would have to be included with the hardware. Software houses would write programs to run under the standard OS and wouldn't have to worry about multiple versions. The source of the OS would be widely available and everyone could make suggestions for its enhancement. No modifications would have to be made to programs to handle I/O. If this had been done in the beginning, an immense amount of effort would have been saved and everyone would be better off. Anyone who thinks it isn't too late to do something along these lines has my enthusiastic support.

For small computers the term "operating system" means any operating system, whether it supports a disk or not. DOS (disk operating system) refers only to those OS's which support a disk device. For IBM 360 & 370 users OS and DOS refer to specific operating systems, both of which support disk. The techniques for doing I/O to a disk and structuring it to keep a directory of files are so complex that very few stand alone programs handle DISK I/O. Altair Disk Basic Versions 4.0-and-before operate this way. All the code for handling the disk is included in BASIC. To support the floppy disks they sell, microcomputer system manufacturers are being forced to develop DOS's. Unfortunately, they are all different so no program will run under two of them. Also, data on floppy disk written under one DOS will not be readable with another DOS. Still, the complexity of the disk forces these manufacturers to write OS's which, if well written, will provide users with the advantages of device independent I/O. Hopefully, non-disk versions of these OS's will be provided to users who can't afford a disk.

I have restricted myself to single-user operating systems. Multi-user OS's, which can run more than one program at a time, are much more complicated. At present, no multi-user OS is available for a microcomputer. Software is the main obstacle preventing microcomputers from handling multiple users since the speed of the hardware is more than sufficient. Digital Equipment has supplied a multi-user operating system for the PDP-8 for over a decade. The PDP-8 has a less powerful and slower instruction set than most microcomputers so there is no reason a powerful time-shared operating system could not be written for microcomputers. It is sure to be done eventually, but probably not for six months to a year from now.

MICROCOMPUTER

PROM'S

H-1024-5B	256 x 4 Bit 70 ns OC	4.25
1702A	256 x 8 Bit 1 us TS Erasable	5.00
1702AL	256 x 8 Bit 1 us TS Eras Lo Pwr	7.00
2704	512 x 8 Bit 450 ns TS Erasable	30.00
2708	1024 x 8 Bit 450,ns TS Erasable	35.00
5203AQ	256 x 8 Bit 1 us TS Erasable	9.00
5204AQ	512 x 8 Bit 1 us TS Erasable	12.00
HM7611-5	256 x 4 Bit 60 ns TS	3.25
74S287	256 x 4 Bit 65 ns TS	9.00
74S387	256 x 4 Bit 65 ns OC	9.00
82S23B	32 x 8 Bit 50 ns OC	4.00
82S129B	256 x 4 Bit 50 ns TS	4.25
8223B	32 x 8 Bit 50 ns OC	4.50

WAVEFORM GENERATOR

8038	VCO	4.50
MC4024	Dual VCO	2.75
566	VCO-Function	2.00

CHARACTER GENERATORS

2513	5x7 5 line CM2140 Upper case	6.75
MC6571	7x9 7 line	12.00
2431-1	CM3421 (Signetics)	5.00

SHIFT REGISTERS

DYNAMIC

1404AN	1024 x 1 Bit 2.5 MHz	3.00
2505K	512 x 1 Bit 2.5 MHz	3.00

STATIC

MM506	100 x 2 Bit	.89
2509K	50 x 2 Bit 1.5 MHz	1.00
2518B	32 x 6 Bit 2.0 MHz	3.95
2533V	1024 x 1 Bit 1.5 MHz	2.00
TMS3002	50 x 2 Bit 1.0 MHz	1.00
TMS3112	32 x 6 Bit 2.0 MHz	3.95
MM5058	1024 x b Bit 1.5 MHz (8 pin)	2.00

MISC OTHER COMPONENTS

NH0025CN	Dual Low Cost MOS Clock Driver	1.75
NH0026CN	5 MHz Dual MOS Clock Driver	3.00
N8T20	Bi-Directional One Shot	4.00
N8T26	Quad Bus Driver/Receiver	3.25
N8T97	Tri State Hex Buffer	1.45
DM8098	Tri State Hex Inverter	1.00
1488	RS232 Quad Line Driver	1.95
1489	RS232 Quad Line Receiver	1.95
D-3207A	Quad NAND to MOS Driver	2.50
C-3404	6 Bit Latch 12 ns O/P Delay	3.95
P-3408A	Tri State Hex MOS Sense Amp	6.75
P-4201	Clock Generator	4.95
MM-5320	T V Camera Sync Generator	6.00
MM-5369	Oscillator Pre-Scaler	2.00
DM-8130	Ten Bit Comparator	2.25
DM-8131	6 Bit Comparator	2.00
DM-8831	4 Input AND NAND Tri State	2.50
DM-8833	Quad Tri State Transceiver (True)	2.50
DM-8835	Quad Tri State Transceiver (Inv)	2.50

U A R T S

AY5-1013	(TR1602A) 30 KHz	6.95

C.P.U. 'S

8008-1	8 Bit CPU	18.75
8080	Improved 8008	22.95
8080A	Super 8008	24.95
Z80	CPU (3880)	39.95
F8	CPU (3850)	29.95

Support Devices

3851	Program Storage Unit (F-8)	19.95
3853	Static Memory Interface (F-8)	19.95
3881	Parallel I/O Controller (Z-80)	19.95
3882	Counter Timer Circuit (Z-80)	19.95
8212	8-Bit I/O Port	4.25
8214	PriorityInterrupt Control	12.95
8216	Bi-Directional Bus Driver	5.25
8224	Clock, Generator & Driver	6.00
8228	System Controller & Bus Driver	9.25
8251	Programmable Communication Int	12.00
8255	Programmable Peripheral Interface	12.00
8257	Progr Direct Mem Access Control	23.00

DYNAMIC RAMS

1103	1024 x 1 Bit 300 ns	1.50
2107A	4096 x 1 Bit 300 ns	6.00
2107B	4096 x 1 Bit 200 ns	6.50
2107B-4	4096 x 1 Bit 270 ns	5.00
2107B-6	4096 x 1 Bit 350 ns	4.50
4008P-6	1024 x 1 Bit 500 ns	2.75
4050NL	4096 x 1 Bit 300 ns	4.00
MM5261	1024 x 1 Bit 400 ns	3.00
MM5262	2048 x 1 Bit 365 ns	3.00
MM5270	4096 x 1 Bit 200 ns (18 pin)	4.00
MM5280	4096 x 1 Bit 200 ns (22 pin)	6.00

STATIC RAMS

21L02-1	1024 x 1 Bit 500 ns TS	2.00
31L01	16 x 4 Bit 110 ns OC	2.00
1101A	256 x 1 Bit 1 us TS	1.00
2101	256 x 4 Bit 1 us TS	3.00
2102	1024 x 1 Bit 1 us TS	1.50
2102-1	1024 x 1 Bit 500 ns TS	1.75
2111A-4	256 x 4 Bit 450 ns TS (18 pin)	4.45
2112A-4	256 x 4 Bit 450 ns TS (16 pin)	2.45
2501B	256 x 1 Bit 1 us	1.45
3107	256 x 1 Bit 80 ns OC	2.95
74C89	16 x 4 Bit 280 ns TS	3.00
74S201	256 x 1 Bit 50 ns TS	4.75
91L02A	1024 x 1 Bit 500 ns TS	2.00
7489	16 x 4 Bit 60 ns OC	2.25
8225	16 x 4 Bit 50 ns OC	1.50
8599	16 x 4 Bit 50 ns TS	1.50

FIFO

3341A	64 x 4 Bit 1.0 MHz	6.75

JADE CO. OFFERS

THE ITEMS SHOWN IN THIS AD FOR IMMEDIATE SHIPMENT FROM STOCK. IN ADDITION WE STOCK OVER 4,000 OTHER ITEMS. INCLUDING:

1. All Standard, Hi-Speed, Low Power, Schottky TTL Product
2. All Linear Devices
3. Transistors—Diodes
4. Clock Kits & Modules
5. Vectors & AP Products
6. Resistors-Capacitors
7. Video Games & Chips
8. PROM Setters [E-Prom]
9. Wire & Wire Wrap Tools

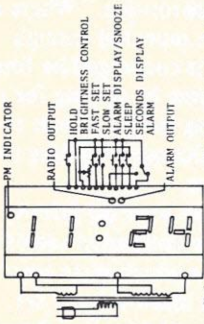
SPECIAL OF THE MONTH

2708 E-PROM
Price $35.00 ea.

master charge

your BANKAMERICARD welcome

Digital Clock Module



FEATURES

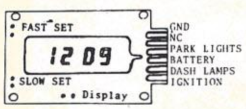
- *4-digit 0.5 display
- *Complete,needs only transformer & switches
- *Clock Radio Alarm
- *12 hr & 60Hz
- *Power Failure Indicator
- *Direct Drive-No RFI

APPLICATIONS

- *Clock radio timers
- *Desk Clocks
- *Instrument panel clocks

MA 1002 A
$10.95

TRANSFORMER,14 VCT AC SCHEMATIC TO POWER MA 1002A $1.75



FEATURES

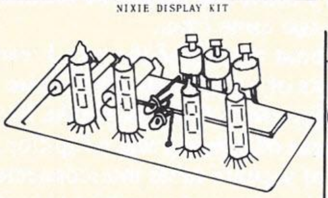
- * Operates on 9-12 V DC
- * 0.3" green display
- * Internal time base
- * Display color filterable to blue, green & yellow
- * Lock out of time set when display is "OFF"

APPLICATIONS

- * In dash autoclocks
- * Recreation vehicles
- * Aircraft/Marine clocks
- * Portable battery operated instruments

MA 1003
$24.95

NIXIE DISPLAY KIT



FEATURES

- * MM5316 Clock Chip
- *Needs only case & 12 V AC @ 200 ma
- *Blanking control
- *AM PM Option
- *24 hr Capability
- *59 minute timer
- *Power Failure indicator
- *24 hr clock radio

NIXIE CLOCK KIT
$12.95
Transformer 110-14V AC 200ma $1.50

TIME BASE CHIP

MM 5369	17 Stage Osc/Divider	2.00

CLOCK CHIPS

MM5312N	4 Digit w/BCD-7 Seg 12/24 hr 50/60 Hz	4.50
MM5316N	4 Digit Alarm-Snooze, 12/24 59 min timer,50/60 Hz,AM/PM	4.00
MM5318N	TV Display Digital Clock	8.95
MM5370N	Gas Discharge Display 12/24 60 Hz, 59 min timer, AM/PM	4.00
MM5375NAC	4/6 digit, date counter, 50Hz	2.00
7001	6 digit calender, alarm, 12/24 hr	5.95

74LS00

Low Power Schottky

74LS00	.40
74LS02	.40
74LS03	.40
74LS04	.40
74LS05	.40
74LS08	.40
74LS10	.40
74LS11	.40
74LS12	.50
74LS20	.40
74LS21	.40
74LS30	.40
74LS32	.40
74LS37	.50
74LS42	.90
74LS51	.40
74LS73	.60
74LS74	.65
74LS76	.60
74LS78	.80
74LS85	2.00
74LS86	.55
74LS95	2.00
74LS107	.55
74LS109	.60
74LS112	.60
74LS113	.60
74LS122	.60
74LS139	1.30
74LS151	1.50
74LS153	1.60
74LS155	1.60
74LS157	1.30
74LS161	1.50
74LS169	2.00
74LS174	1.25
74LS175	1.30
74LS191	2.45
74LS192	1.95
74LS193	2.45
74LS196	2.60
74LS248	1.35
74LS251	2.00
74LS266	.75
74LS363	.70
74LS367	1.00
74LS368	1.00
74LS386	.75

CMOS

34001	Quad 2-Input NOR Gate	.40
4000	Dual 3-Input NOR Gate/Inver	.25
4001	Quad 2-Input NOR Gate	.25
4002	Dual 4-Input NOR Gate	.25
4004	7 Stage RPL Ctr/hz Div.	3.50
4006	18-Bit St. Shift Register	1.40
4007	Dual Comp Pair + Inverter	.25
4008	4-Bit Full Adder	1.25
4009	Hex Buffer Inverter	.70
4010	Hex Buffer Converter	.70
4011	Quad 2-Input NAND Gate	.25
4012	Dual 4-Input NAND Gate	.25
4013	Dual Type D Flip Flop	.60
4014	8-Bit St. Shift Register	1.25
4015	Dual 4-Bit Static Shift Reg	1.25
4016	Quad Analog SW/Quad Multi	.60
4017	Decade Counter/Divider	1.25
4018	Divide By N Counter	1.25
4019	Quad AND-OR select Gate	.70
4020	14-Bit Binary Counter	1.25
4021	8-Bit St. Shift Register	1.25
4022	Octal Counter/Divider	1.25
4023	Triple 3-Input NAND Gate	.35
4024	7-Stage Ripple Counter	1.00
4025	Triple 3-Input NOR Gate	.35
4026	Decade Ctr/Div 7 Seg Output	2.25
4027	Dual J-K Flip Flop	.60
4028	BCD to Dec.Bi.Octal Dec	1.25
4029	4-Bit BIN/BCD UP/DN CTR	1.90
4030	Quad Exclusive OR Gate	.60
4032	Triple Serial Adder (POS)	1.60
4033	Decade Ctr/Div 7 Seg Out K/B	2.00
4034	8-Bit Universal Bus Register	3.50
4035	4-Bit Shift Register	1.60
4038	Triple Serial Adder (Neg)	1.60
4040	12-Bit Binary Counter	1.50
4041	Quad True/Comp Buffer	1.45
4042	Quad Latch	1.25
4043	Quad 3-State NOR R/S Latch	1.20
4044	Quad 3-State NAND R/S Latch	1.25
4046	Phase-Locked Loop	1.95
4047	Mono/Astable Mltvbr	2.50
4049	Hex Inverter/Buffer	.75
4050	Hex Non Inverting Buffer	.75
4051	8 Channel Analog Multiplexer	1.10
[illegible]	Dif 4 Channel Analog Mx.	1.10
4053	Triple 2 Channel Mltplx	1.10
4060	14 Stage Binary Ctr/Osc	3.25
4061	256 x 1 Static RAM	7.00
4063	4-Bit Magnitude Comparator	2.50
4066	Quad Bilateral Sw.	.85
4068	8-Input NAND Gate	.38
4069	Hex Inverter	.40
4070	Quad Excl OR Gate	.38
4071	Quad 2 Input OR Gate	.38
4072	Dual 4 Input OR Gate	.38
4073	Triple 3 Input AND Gate	.38
4075	Triple 3 Input OR Gate	.38
4076	Quad D Type Register	1.85
4078	8-Input NOR Gate	.38
4081	Quad 2 Input AND Gate	.38
4082	Dual 4-Input AND Gate	.38
4093	Quad 2-1 NAND Schmitt Trigger	.75
4098	Dual Retrig/Reset M S Multi	2.50
4160	Decade Counter	1.75
4161	Binary Counter	1.75
4162	Decade Counter	1.75
4163	Binary Counter	1.75
4174	Hex D Flip-Flop	1.75
4175	Quad D Flip-Flop	1.60
4194	4-Bit Univ Shift Register	1.80
4501	Triple Gate	.38
4502	Strobed Hex Inverter/Buffer	1.75
4503	Hex Three-State Buffer	1.15
4506	Dual Expandable AOI Gate	.70
4507	Quad Exclusive OR Gate	1.00
4508	Dual 4-Bit Latch	4.00
4510	BCD Up/Down Counter	1.75
4511	BCD to 7 Seg Latch/Decdr/Dr	2.00
4512	8 Channel data selector	1.50
4514	4/16 line decoder (hi)	4.75
4515	4/16 line decoder (lo)	4.75
4516	Binary up/down counter	1.75
4517	Dual 64-Bit Static Shift Reg	8.50
4518	Dual 4-Bit Decade Counter	1.65
4519	4-Bit AND/OR Selector	.90
4520	Dual 4-Bit Binary Counter	1.65
4521	24 State Frequency Divider	3.25
4522	BCD Divide by N-Counter	1.75
4527	BCD Rate Multiplier	3.00
4528	Dual Monostable Mltvbr	1.75

CONNECTORS

EDGE CONNECTORS

10	Dual .156 W W Gold	.50
15	Dual .156 W W Gold	.75
22	Dual .156 Solder Gold	2.25
50	Dual .125 Solder Gold	5.00
50	Dual .125 Solder Gold (No Bolt Down Ears)	3.50
10	Molex type w/edge clip female & male (solder)	10/1.25

SUBMINITURE

25	Solder type (25-S) (Female)	3.90
25	Solder type (25-P) (Male)	3.50
	Male & female (25-P & 25-S)	6.50
	Cover each	1.50

JADE CO

Electronics for the Hobbiest and Experimenter

5351 WEST 144th STREET
LAWNDALE, CALIFORNIA 90260
(213) 679-3313

SOCKETS

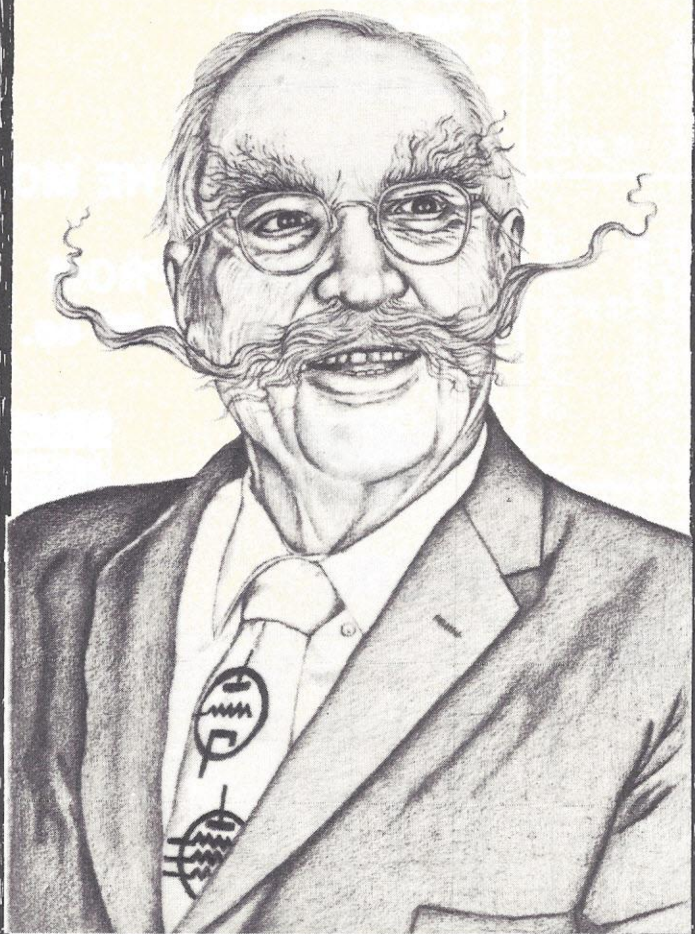
I C SOCKETS

Pins	Description	Price
8	Lo Pro Tin	.20
10	Transistor IC-10) Gold	.25
14	Lo Pro Tin	.25
14	Wire Wrap Gold	.30
16	Lo Pro Tin	.25
16	Standard P C Tin	.30
16	Wire Wrap Tin	.30
18	Lo Pro Tin	.25
22	Lo Pro Open Frame Tin	.50
22	Wire Wrap Tin	.95
24	Standard P C Tin	.50
24	Wire Wrap Tin	.85
28	Lo Pro (Open Frame) Tin	.40
28	Standard P C Tin	.60
28	Wire Wrap Tin	.95
40	Lo Pro (Open Frame) Tin	.50
40	Wire Wrap Tin	1.45

CIRCLE 17

I REMEMBER BESSIE

by Henry Brainerd



The Mark I (nicknamed Bessie), Eniac, Whirlwind and IBM 650 are all but forgotten today, but in the 40's and 50's they began the computer age.

In the summer of 1944 I started a new job at Harvard, developing military equipment for voice communications. What's that got to do with computers? Only this: Across the hall was a strange new monster, dedicated the day after I started work. I got in literally by the back door, wangling permission to bring my nickels and those of my peers to their Coke machine.

They had a big room, like two classrooms thrown together, with the machine filling one entire wall. They – the people – were friendly enough, though preoccupied with their jobs. Howard Aiken was the boss – after all the critter was his brainchild. The others I remember are Grace Hopper, on leave from Vassar, and a Campbell whose first name I forget. They were kind enough to explain the machine to me as long as I was careful not to ask for classified information.

The machine used 24-digit numbers – decimal of course – which were stored in several dozen registers somewhat like automobile odometers. These registers were also adders – whatever number was delivered to one was added onto the previous contents in the unbelievably fast time of a fifth of a second. Multiplying was done in a separate unit in four-fifths of a second, while dividing took another unit all of four seconds. Newspapers reporting the dedication commented on the tremendous speed.

There were a large panel of dial switches for entering constants, an electric typewriter, a card punch to deliver the results and a reel of wide punched paper tape that programmed the entire operation.

The machine was based on a requirements that Prof. Aiken set up for scientific calculations. IBM built the machine and donated it to Harvard. By that time the war was on. Harvard leased the machine to the Navy, which commissioned Aiken, Hopper and Campbell as Commander, Lieutenant Command and Lieutenant.

My job terminated at the end of the war, and I pounded the pavement for almost a year before I found an opening in the MIT Instrumentation Lab. Meanwhile I saw occasional news items about Aiken's baby. He found that where a sequence was repeated often in a program it could be put on a separate loop of control tape which was called in as often as needed. This came to be called a "subroutine." Where a computation was repeated enough times to be sure it converged he found you could save time by testing for residual error and as soon as it was reduced to insignificance returning to the main program.

Whenever there were no priority jobs, the machine computed tables of Bessel's functions which were eventually published in a multi-volume set. That's where the machine's nickname of Bessie came from.

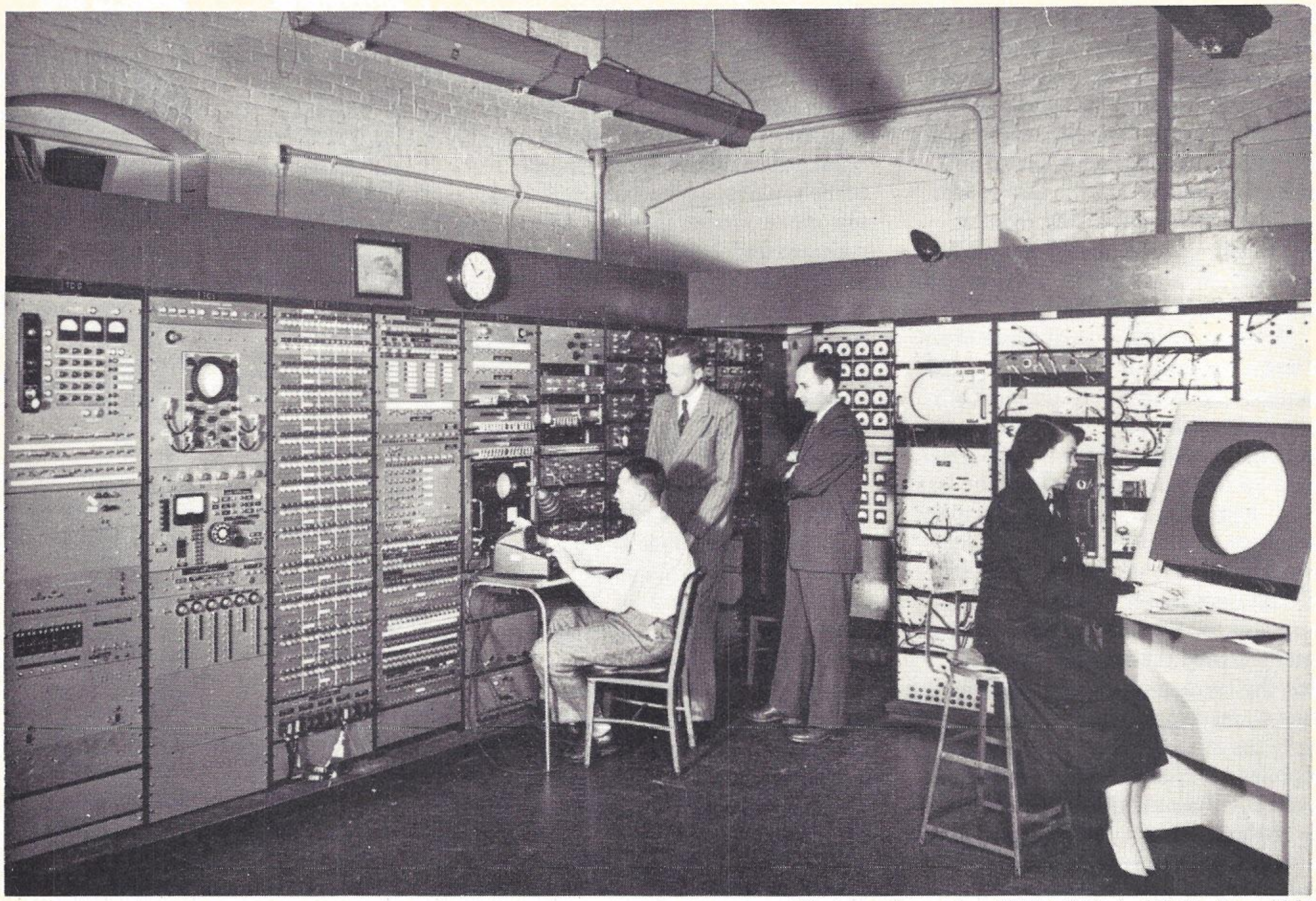
About the end of the war, I read reports of a computer called Eniac at the University of Pennsylvania. Each element of memory was a flip-flop, a pair of vacuum tubes interconnected so that making one conduct turned off the other. Being all electronic, it was much faster than Bessie, with speeds measured in milliseconds or even (believe it or not) in microseconds.

On my new job – technical writing – my first assignment was a report on a big job of computing that had just been completed. The work was done by a roomful of girls punching desk calculators and writing the numbers on big sheets of paper.

Meanwhile the publications group that handled my work was also han-

dling progress reports from Project Whirlwind. This was a daring new all-electronic computer that was planned to be speedy enough to simulate the performance of, say, an airplane, as fast as life. It could include in its loop a human pilot at the simulated controls. The goal was said to be a million additions a second. It would operate in that strange new medium, binary arithmetic, totally foreign to decimally minded people.

As the months went by, the sub-



Photograph courtesy M.I.T. Historical Collections 1951

BIG BRAIN – This is the operational center for Whirlwind 1, the high-speed electronic computer at the Massachusetts Institute of Technology. Watching solution of a problem are, left to right, Stephen H. Dodd Jr., in charge of the computer operation; Jay W. Forrester, director of the laboratory; Robert R. Everett, associate director; and Ramona D. Ferenz

assemblies were being designed, built, then at last tested. I remember particularly that the memory was an array of cathode ray tubes, as many as there were binary digits in the computer word. Each tube had a raster of spots, with memory by the presence or absence of charge on a spot.

Eventually Whirlwind was actually operated. The machine with its staff filled a fair-sized two-story building, and I rather suspect that the electronics took more space than the people.

Even when Bessie was young we began to hear that having a computer wasn't enough; telling it what to do was the big chore. Bessie's storage registers were also adders, so people sometimes forgot to reset a register to 0 and instead added onto whatever was already there. Whirlwind had magnetic tape, introduced since the war. You coded your program on that, then you tried it out. Always you made a few mistakes, so you coded short correction tapes, produced a new tape and after a while had a program that worked. Of course, finding the mistake meant running the machine to a stop where you thought the mistake was, then stepping ahead one instruction at a time until you found where it went wrong. This tied up the whole machine for an hour or so.

One summer the Whirlwind staff offered a really innovative course in programming. They wrote programs that simulated two hypothetical machines with different instruction sets, so that you could learn how to program three different machines.

Meanwhile back at the Lab we still had our roomful of young women punching desk calculators. Sometime in the 50's we took a step forward with the CPC, the Card Programmed Calculator. This machine took a data card and an instruction card and performed arithmetic electronically, much faster than a girl with a desk calculator and with less chance for mistakes. I enjoyed watching the flashing neon lights, but I never really understood it. I suppose the output was more punched cards.

Then about a year later we took the big step, got an IBM 650 computer. As far as I know, the 650 was the first full-fledged computer produced on a commercial scale, and ours was the first one off the production line. It was displayed in the main entrance lobby of MIT, then installed in the room we had made ready for it with a false floor that cables could run under and special air conditioning.

Of course we all crowded in to stare at our own 650. It was unbelievably complex, said to have 2000 amplifier vacuum tubes and 6000 vacuum diodes. The machine was housed in three cabinets, each taking roughly 2 by 6 feet of floor space, connected by cables under the false floor. One cabinet had the card reader and punch, which were the only input and output. One had the main memory – a high speed drum – and one had the electronics for arithmetic and control. In addition there was a printer fed by the cards

and there were the usual keypunches, verifiers, sorters and the like.

Instruction manuals were freely available and made the machine sound almost simple. It had a word of 10 decimal digits. Arithmetic was bi-quinary: a quinary part with values 0 to 4 and a binary part valued 0 or 5. Data was stored, however, in binary-coded decimal form, four bits per digit, and was automatically transformed to bi-quinary as it was brought into the arithmetic unit, back again when it was stored. The arithmetic unit, as far as the user was concerned, was a 10-digit "distributor" and a 20-digit "accumulator." The accumulator held one of the operands, then received the result of any arithmetic operation. The distributor held the last word received from or sent to storage. The distributor and each half of the accumulator could also be addressed independently.

Memory, as I have said, was a drum. It had a capacity of 2000 words in 20 bands of 50 words each to hold both data and instructions. Addresses were numerical, 0000 to 0049 for the first band up to 1950 and 1999 for the last band. Instructions were first two digits operation code; next four digits, operand address; last four digits, address of next instruction. Having each instruction tell where the next was located allowed them to be placed so that the turning of the drum during the time an instruction was executed would bring the next instruction under the read head without delay. This was theoretically a beautiful way to optimize throughput but in practice was an unmitigated pain. Putting the instructions in the same memory with data allowed indexing by adding to the data address in an instruction.

The 650 is almost forgotten today. I have even seen historical articles that seem to be unaware of it, going directly from Bessie, Whirlwind and Eniac to the IBM 700 series. To me it seems that the 650 was to computers what Henry Ford's Model T was to automobiles. As a railroad hobbyist I was much aware of how railroads made rapid use of this new machine. They used it for everything from routine accounting to structural design of bridges to optimizing schedules so as to get a maximum number of trains over a busy track. The 650 was important in the development of automation in the railroads.

We must have kept the 650 nearly 10 years. On the one hand we added new features as they became available — alphanumerics, indexing accumulators, floating point arithmetic, a tape drive, a small section of core. Each of these meant another big cabinet to put in that increasingly crowded room.

On the other hand we started experimenting with memoric coding. Our first coding system was called Mitilac for Massachusetts Institute of Technology Instrumentation Laboratory Automatic Coding, but my boss suggested that it sounded like a new brand of canned baby formula. It used a three-address code, something like

396 P 407 214

which would mean contents of 0396 plus contents of 0407 store in 0214. It provided for arithmetic operations, trigonometric functions, conditional branch, input, output and perhaps a few more. I remember that I wrote one program under this system. It computed the fundamental and second harmonic and the linear drift of data from a particular instrument test. The capacity of the machine was so limited that I had to compute and subtract out the sinusoids, punch out the remainder, then start a second section and feed it the output of the first.

Several members of the computer staff tried their hands at what we would now call assembly language: Flad, Baltitac, Slic. Flad stood for FLoating ADdress. I don't remember what the others stood for; acronyms were a dime a dozen. You used alphanumeric symbols for data quantities and machine operations, then the assembly assigned addresses, producing a deck in machine code that you could load whenever you ran your program.

The biggest step was what is now called a high-level language, Mac. It was said to stand for Massive Algebraic Computation. The thought was that any engineer could use Mac to write his own programs, and the problem of people to do programming would be solved once and for all. It didn't work quite that way. There were still quite a few constraints for the user to master, and debugging tried a man's patience to the breaking point. You had to develop a computer outlook, or as I once put it, "Programming a computer is like training a dog: you have to know more than the dog."

Most of us didn't know much more than the computing dogs we were trying to put to work, and 30 years later many of the same old problems remain. When I see engineers struggling to outwit their wonderful new microcomputers, it surely does remind me of Bessie.

When is a computer not a computer? When it's contraband.

UNDERCOVER COMPUTER

by Henry Gilroy

Nameless Jones is a bright, successful young man who has made his mark as a scientist and is now being pushed rapidly up the management ladder in a big company. When he issues instructions, a couple of hundred people pay close attention, because he is already manager of a whole corporate division. His memos are studied carefully by lesser folk who must fathom the strange workings of his mind to do his will. His financial decisions affect the ebb and flow of hundreds of thousands of dollars at a time. A parking space is reserved for his use. Surely he is becoming a man of some importance.

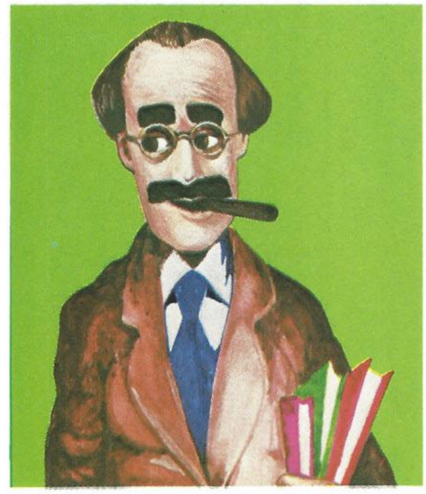
He cannot, however, obtain a personal computer for use in his office, though he urgently desires to do so. He's not lacking in computer services, of course, only in access to an independent system under his full personal control. He now has a computer terminal in his office that connects him to a great big 370-something at company headquarters in the next state. He can (and does) tap into the reservoir of computing facilities at headquarters, picking software from a huge library, pulling company data of importance from a continuously updated fund of information. His terminal is fast and fairly easy for him to use.

Unluckily, he can't always make the system do what he wants it to. He can't try a number of little technical ideas of his own, because the system isn't set up to respond to his whims, only to do what the computer folks at headquarters think a computer *should* do. He may be able to persuade them to give him extra help from time to time, but it's a problem for him to set up meetings and prepare a convincing story that will enlist their enthusiastic support. Since he doesn't speak their special language very well, he always feels slightly inferior and embarrassed when he talks to them.

Nameless does use the terminal as much as he can, since his division is being billed some $50,000 a year for the thing, and he hates to see the money go to waste.

He's never actually seen the computer, either. Once, when he was at a headquarters meeting, he managed to get into the computer department with the intention of finding out what sort of machine lived at the other end of the wire leading from his office. After he had taken two paces into the computer room, a pair of fellows grabbed him and ejected him physically from the premises. He wasn't entirely sure why, but apparently he hadn't made an appointment and wasn't being escorted by the right sort of people. He explained about the $50,000 a year and his wish to chat with somebody about how the system might be adapted to his special needs in some small way, but nobody in the computer department was interested in his problems.

Recently, Nameless has been admiring personal computer systems in the magazines and at a local computer store. He is greatly excited by the sort of system he could buy for perhaps $20,000 – loaded with RAM, Prom, floppies, printers and color CRT displays. For a very few thousand dollars more, he could interface that little system to some of his experiments. At no extra charge, he could arrange for his secretary to use the system for word processing several hours a day. He could run experimental budgets



and project plans without the sneaking hunch that other managers in the company, competing for budget money (including the manager of the computer department itself), were studying his tentative plans and figuring out how to counter him. At lunch he could play Star Trek, too. A world of wonders would be opened to him by a microcomputer system that would also save the firm a rather large sum of money.

He tried unsuccessfully to buy such a system. The company had been hypnotized by its computer department, which persuaded top management that no purchase of data processing equipment or services should be allowed without their approval. Their argument is good, even sensible, in a world that does not change in surprising ways. They know and can show that corporate computer operations get completely out of control unless firm stands are taken to prevent every Tom, Dick and Harry in the company from doing whatever he wants. Programming and equipment are duplicated, even triplicated, if the computer overlords don't prevent these evils from happening. Efficiency is ruined. Money and time are wasted. The company needs a single, coordinated approach to computer usage. Computers are complex and expensive. Programming is mysterious and difficult. Only highly trained professionals can be ex-

CIRCLE 19 ▶

The POLY 88 Microcomputer System

PolyMorphic Systems now offers the complete, assembled, personal computer system—the POLY 88 System 16. A full 16K system with high speed video display, alphanumeric keyboard, and cassette program storage. A BASIC software package providing the most advanced features available in the personal computing market. Features like PLOT and TIME, which utilize our video graphics and real-time clock. Others like VERIFY, so that you know your tape is good before you load another. Or input type-ahead so you can tell your program to run while the tape is still loading (it stores up to 64 characters of commands or question responses to be executed). All these plus a complete package of scientific functions, formatting options, and string capabilities. With the POLY 88 System 16 you can amaze your timesharing friends the very first night!

PolyMorphic Systems 11K BASIC

Size: 11K bytes.
Scientific Functions: Sine, cosine, log, exponential, square root, random number, x to the y power.
Formatted Output • Multi-line Function Definition • String Manipulation and String Functions • Real-Time Clock • Point-Plotting on Video Display • Array dimensions limited by memory • Cassette Save and Load of Names Programs • Multiple Statements per Line • Renumber • Memory Load and Store • 8080 Input and Output • If Then Else • Input type-ahead.

Commands: RUN, LIST, SCR, CLEAR, REN, CONT.

Statements: LET, IF, THEN, ELSE, FOR, NEXT, GOTO, ON, EXIT, STOP, END, REM, READ, DATA, RESTORE, INPUT, GOSUB, RETURN, PRINT, POKE, OUT.

Built in Functions: FREE, ABS, SGN, INT, LEN, CHR$, VAL, STR$, ASC, SIN, COS, RND, LOG, TIME, WAIT, EXP, SQRT, CALL, PEEK, INP, PLOT.

Systems Available. The POLY 88 is available in either kit or assembled form. It is suggested that kits be attempted only be persons familiar with digital circuitry. The following are two of the systems available.

System 2: is a kit consisting of the POLY 88 chassis, CPU, video circuit card, and cassette interface. Requires keyboard, TV monitor, and cassette recorder for operation. $690

System 16: consists of an assembled and tested System 2 with 16K of memory, keyboard, TV monitor, cassette recorder, 11K BASIC and Assembler on cassette tapes. $1995.

Prices and Specifications subject to change without notice.
California residents add 6% sales tax.

PolyMorphic Systems
460 Ward Drive Santa Barbara, Ca. 93111 (805) 967-2351

pected to understand computer operations, making correct judgements on acquisitions and allocations of hardware and services.

Nameless is not a trained computer scientist, merely an upstart biologist who thinks he can abuse his computing privileges because he has the political power of a division manager. Hell will freeze over long before the computer department allows Nameless to have his own computer. In fact, the department even forces a large company division in South America to work on-line with the computer at headquarters in Oklahoma via satellite rather than let them have a small system of their own. The department has sent back to South America several communications that were written inadvertently in Portuguese – with good, stiff reprimands pointing out the proper procedures for communicating with Oklahoma in English.

"But Nameless," you may say. "The world *does* change in surprising ways. Why don't you buy the computer and call it something else?" Easier said than done. The folks in the computer department are intelligent, sincere, and in most respects even *right.* They will detect computer purchases unless they are extraordinarily well-concealed. Nameless needs wise assistance in keeping his computer undercover. A number of suggestions have already been offered, some of which will be recited here. Before Nameless takes the plunge and starts issuing purchase orders, perhaps PERSONAL COMPUTING readers can suggest covers and strategies that will help him and others like him in the grim grasp of entrenched computer bureaucracy.

One ploy suggested is to identify the computer as a calculator. Calculators are typically beneath the dignity of the computer department and may be purchased without its clearance. A really good programmable calculator costs as much as an assembled personal microcomputer without many optional extras. In the past, identical computers have been sold to aerospace commercial customers as computers and to school systems as calculators. Only the model numbers were changed to protect the crafty.

Nameless is enough of an elektroniker to assemble a kit readily, so he might purchase an "electronic parts kit." The purchasing agent might wonder what it's for, but at least it would not come to the attention of the dreaded computer department. Actually, if Nameless were enough of an elektroniker, he could buy IC components and build up a system from scratch without much trouble. Indeed, he could purchase rather large systems like electronic games, complete with video displays, that would allow him to have a significant head start on construction of a real computer system. (Somebody suggested buying a washing machine or a microwave oven with a microprocessor controller, then throwing away everything but the microprocessor and power supply. That seems a bit extreme.)

Word-processing systems and report generating equipment are not always identified as computers. Under the right conditions a clever buyer ought to be able to disguise a computer as a word processor with lots of RAM, a video display, a Diablo or Qume, and twin floppies, never mentioning the word computer.

Manufacturers of computers are not all set up to play this game and may not be able to supply literature that says the right things on it. This provides an opportunity for some entrepreneur to offer an important service.

Nameless needs a supplier who is prepared to sell him anything he wants, complete with proper printed literature cast in the right terminology. This supplier could arrange to purchase computer equipment from manufacturers and distributors – perhaps on an OEM basis – and rework it into products that Nameless could buy.

For example, Nameless may ascertain that his division urgently needs an oven for incubating experimental eggs. He'd call his Special Supply Co. and ask if it can provide an incubator with certain automatic features. He might ask for temperature and humidity sensors that convert analog data from the incubator to digital form and feed it to an automatic control system that analyzes the performance of the incubator, types out reports of that performance and feeds in programmed instructions to control future performance. The operator of the system must be able to enter programmed instructions, to display any of the recorded data in convenient form and print out reports automatically.

The supplier would assemble exactly this system, choosing a basic incubator that is modest in price (perhaps a cardboard box with an electric light) but supporting it with a first-rate microcomputer system that does everything Nameless specifies and everything else he might wish for.

The supplier would prepare data sheets, devise model numbers, prepare all the covering information in formal, reassuring form (probably using a word-processing system of his own) and offer his proposal to the purchasing agent Nameless designates. Chances are that no other company can offer a system that meets the specifications quite so nicely and a sale will be consummated.

Then again, Nameless may want a special copying machine, a coffee-maker (what with present coffee prices, significant savings might be effected by a precisely controlled coffee brewing system), a refrigeration unit, a water purification unit, a laundry installation, numerically controlled postal scales (stamps are expensive now, too), security systems, cafeteria efficiency monitors, atmospheric pollution monitors, janitorial service scheduling machines, report page collators, precise timers, fishtank controls, parking lot traffic guidance devices, analyzers, instruction set generators. Control and measurement systems of all sorts are likely candidates.

In government organizations, because certain activities along these lines are illegal as well as unusual, care must be taken by people like Nameless to avoid trespasses against the law, else they will do time when they are caught.

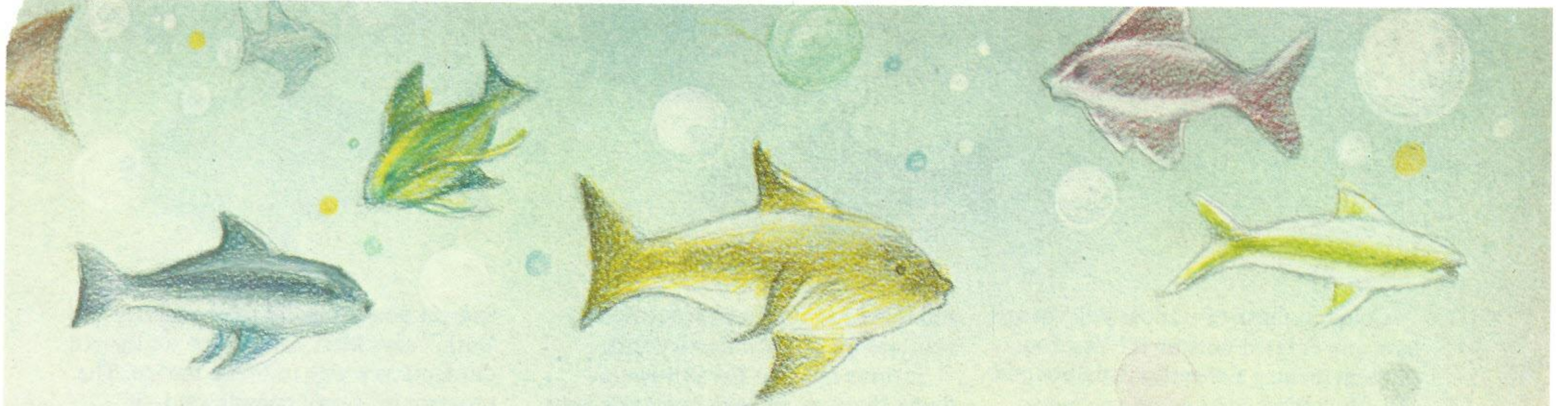
And they *will* be caught. No question. The computer department people are smart and hardworking. Sooner or later they'll begin to notice that Mits is on the list of suppliers to whom the company is issuing checks for purchases. If they know that they haven't authorized purchase of any Mits equipment, this will strike them as odd. Why would the company buy supplies and small parts for stuff they don't have? Aha!

But by then, Nameless may have had his system up for a year and be operating so handily that higher level management will be unwilling to shut him down. Indeed, they may seek his advice on techniques for outflanking the computer department empire builders and getting handy computers for themselves.

If enough good suggestions have been provided to him, Nameless may be able to give them significant help.

DOTS break the ice

The premise of the Lemonade Computer Service Company is that almost anyone can put off complaints that he is "just fooling around with an expensive toy" by going into business. This concept is compatible with our tradition that paper routes and lemonade stands are worthwhile institutions. The Lemonade Entrepreneur stands to make a buck if he applies his computer to commerce – as this article shows.



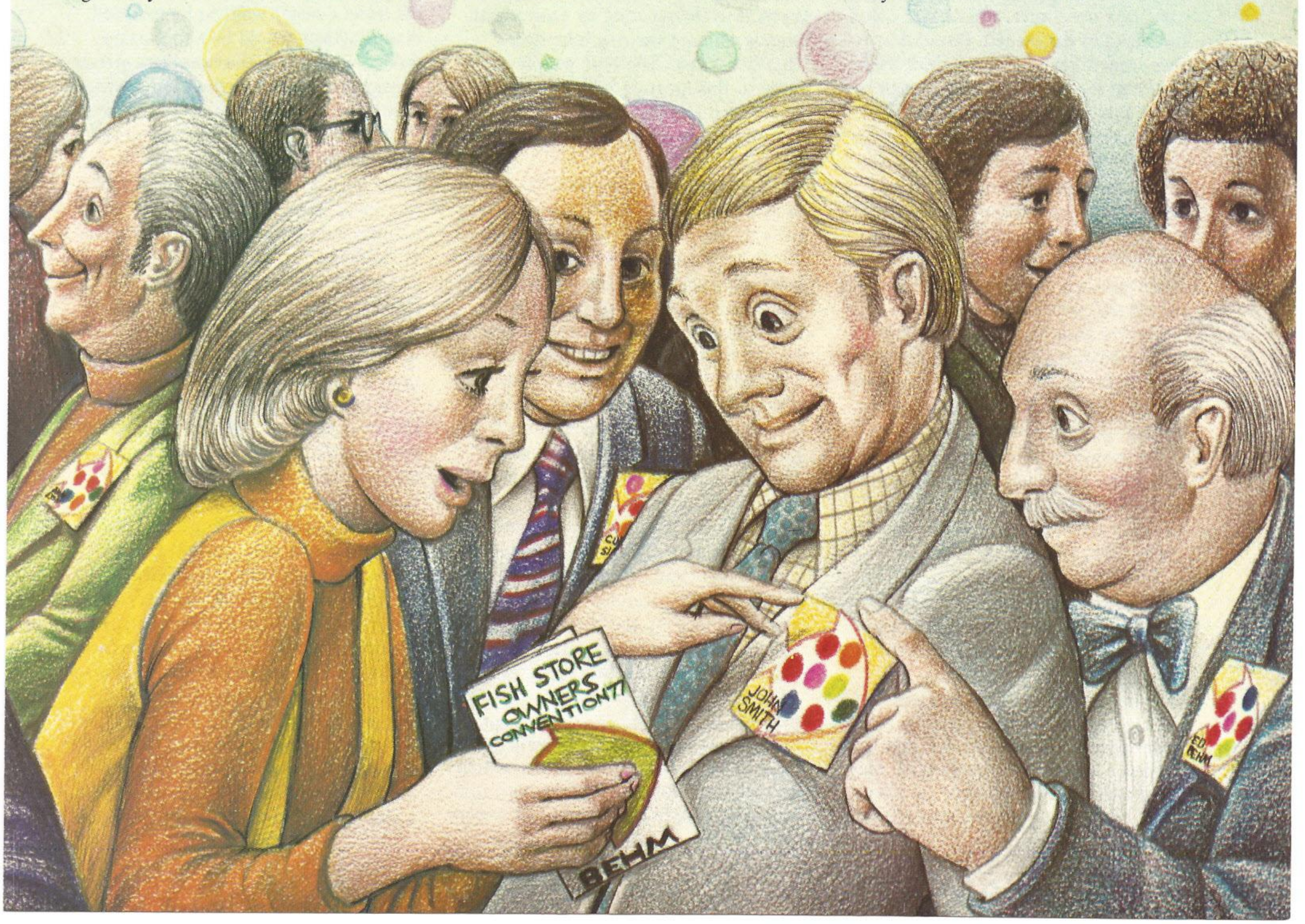
You have just driven 240 miles to a strange city, where the difficult traffic rasps your nerves while you get lost several times looking for the hotel. You're checked in, now, after some confusion about the reservations and half an hour of making phone calls to get the problem straightened out. You have put your suitcase in your room, washed off the travel grime, taken a 10 minute breather while digging the papers out of your briefcase and decided to go down to the convention registration desk to sign in.

When the elevator doors open in front of you, you look into the hotel lobby and see a crowd of uneasy-looking strangers, all handing papers back and forth to each other or standing gloomily around the entrance to the convention meeting room. A cold feeling of despondency drops over you like a wet blanket and you realize that you are sorry you have come. If you hadn't already invested a lot of time and money in the trip, you'd go right back up for your suitcase and leave for home. Ah well, you can't just hide in the elevator. Go and get registered.

The people at the desk are friendly enough but busy with a pile of paperwork. After some fuss, they hand you a program and an identification badge. You pin the badge to your lapel, worrying vaguely about whether the mark of the pin will stay in the cloth forever, and turn toward the doorway of the meeting hall where you stand for a couple of minutes, feeling lonely and miserable.

You notice after a short while that there's something odd about the identification badges other people are wearing. The badges bear not only the name of the individual and the symbol of the organization whose convention you are attending, the Tropical Fishstore Operators Assoc., but an odd little cluster of colored dots. Without actually staring, you manage to get a close look at the badges of people nearby so you can figure out what the dots mean.

You can't tell what they mean. They aren't labeled, and they are in no apparent order. A few people have as many as six colored dots on their badges — red, white, blue, green, orange and yellow. Others have only two or three dots and at least one passerby only one. Why?



Come to think of it, does *your* badge have any colored dots on it? Yes, four of them making a cheerful little bouquet. But why? What did you do to deserve colored dots? You begin to drift through the crowd slowly, listening for snatches of conversation that will give you some hint to the dot-meaning. Sure enough, people are chattering about the dots here and there, trying to figure out their scheme of distribution. When you reach two people who are talking about blue dots, you stop and edge closer, since you have a blue dot yourself.

Soon, one of the speakers notices you and your blue dot and asks you if you know what it's all about. You don't, of course, but you introduce yourself to these blue people and join their conversation. Somebody remembers that the pre-registration papers distributed to all prospective attendees of this meeting three months earlier included a brief questionnaire that asked some odd questions . . . like hair color and make of car owned. Could that have something to do with this? Well, your hair is brown and so is that of one of the other blues. But the third is bald, with a white fringe. Can't be hair color. Cars? Pretty soon you know what kinds of cars these people drive and they know about your hotrod, but that isn't the common factor. You talk about families, your years in the business, other meetings you've attended, mutual friends and your home cities.

As you chat happily, other blues join in and a couple of reds who notice the red dot on your badge. This group breaks up as the formal meetings begin, but you find as the day goes on that strangers with dots like yours stop you in the hallways, restrooms, elevators and restaurants to discuss what you may have in common that is symbolized by the dots.

Red turns out to be a symbol for your home state. Some of the reds are not only from your state but from your city. Your orange dot, you discover the next day, shows that you have contributed at least one article to the organization's magazine, the *Fancy Fish Fancier.* When you learn this, you suddenly notice fellow writers all over the place. Well, not *all* over the place, actually. You are rather a special group, and it is good fun to be identified by your orange dot as a literary sort.

It turns out that the yellow dot shows that you've been operating your store for 10 years or more. Your personal dots are now all cleared up, except for the blue. The blue remains a total mystery until you enter a meeting room on the second day to attend a discussion to which you have been looking forward with especial interest – and you realize that almost everybody in the room is marked with a blue dot. Aha!

You are not the first to notice. Animated conversations are under way all over the room and you join in at once with a group that is grilling two cheerful people who do *not* sport blue dots. What sets them apart? At last it comes out. They are not really very much interested in the topic of this scheduled discussion, *The Gourami: Will it or Won't it?* They were merely dragged to the meeting by enthusiastic friends who are wearing blue dots. That's it at last! The blue people all came to this meeting with an overpowering interest in the gourami question. Blues are a brotherhood of gourami nuts.

When you leave the hotel for the horrible drive back home after three days at the meeting, your heart is light. You have a dozen new friends and your briefcase bulges with the names and addresses of gourami nuts, enough to keep you busy with correspondence for the next year. Marvelous! You even wear your badge all the way home, as if you had absent-mindedly forgotten to remove it, so that when you get home, somebody will ask you about the dots and you can tell them all about it. And next year . . .

Where did the dots come from? Well, the computer program that manipulates the statistics and assigns dot selections to individual conference registrants was developed by the cheerful band of gaming and simulation experts in the Comex group at the University of Southern California in Los Angeles. Their business involves them in a great many meetings at which ill-at-ease strangers from distant cities gather to share their knowledge. "We noticed how hard it is for many people to relax with strangers and get down quickly to productive work," says Mark James, "so we figured out Dots as a way to break the ice. The program isn't very complicated, of course, just a simple routine that almost anybody can do."

The Dots program was tried with great success at several meetings and is now used as a matter of routine when the occasion is appropriate. So, the tested program came from Comex.

The Dot system at any particular meeting you may attend in the future will probably be produced by some entrepreneur who recognizes in Dots an opportunity to set up a Lemonade Computer Service Company enterprise that will let him use his personal computer to make a few dollars.

Every Lemonade Entrepreneur can figure out how to visit the convention manager of a nearby hotel to offer his Dots Production Service for the use of any meeting coming to the hotel. If the hotel doesn't want to be involved directly (Dots would be a nice extra service with which the hotel can attract convention business), then the manager may steer the Lemonade Entrepreneur to the decorator who supplies equipment to the conventions or the travel agency that books conventions or even to the organizations that are holding the conventions. The Chamber of Commerce in every town or the Tourist Commission usually knows what conventions and meetings are coming up and may be willing to give the Lemonade Entrepreneur a push in the right direction.

A personal computer owner with this Dots program and a little common sense should be able to provide questionnaires to the organization holding the meeting, extract data from those forms when they are returned, process the data in his computer system and provide to the customer a detailed specification of the badges by number, name and dot color. Indeed, a thoughtful operator will probably even provide the customer with numbered badges already printed and dotted so they need only be handed out to registrants.

What's a reasonable price for this service? That is left to you Lemonade People in your business wisdom. Clearly, a Dot Production Service is a practical

business project like any other and you'll want to do your homework carefully, identify your suppliers, size up the job and make your price schedule fit the facts. (For some assistance, see Lemonade Planning for Sale by Glenn R. Norris, PERSONAL COMPUTING, March/April 1977).

The Dot program listing accompanies this article. Note that it's written in Fortran, to be run on a big system, but can readily be adapted to small systems using Basic.

The Dot-master who set up the badges for the Tropical Fishstore Operators Assoc. would have begun work with that organization some months ahead of the meeting, probably after demonstrating a simple Dots exercise to the board of directors. He'd have helped with the brief questionnaire being sent to all prospective conference registrants, seeing to it that several handy questions were included among the many that the organization might normally ask.

For example, he'd have made a real point of asking which topic of the meeting would be of greatest interest to attendees, presumably so that adequate seating and P.A. system arrangements could be made. The respondents could choose among:

- Ick Remedies
- The Guppy Bonanza
- The Gourami: Will It or Won't it?
- Keeping the Fish Alive Until the Customer Leaves the Store
- All that Glitters is not Goldfish

The inquiry about the type of car the respondent drives might have been handled simply by asking for the brand of car, rather than offering multiple choices. It's easy enough to go through the responses afterward to list all the answers and let the computer tally them by type. The Dot-master is searching for groupings of special interest to the conference attendees. He'd be happy to discover that 10% of the respondents ride motorcycles or horses instead of driving cars. Perhaps a significant group drives custom vehicles. While there is no great interest in driving a Ford or a Chevy, driving *yellow* Fords, Chevvies, or whatever, may be the entertaining basis for fellowship.

Hair may be classed thus: Blond, Brunette, Red, Bald and Wig-wearing.

This fishstore meeting was regional, drawing attendance from only half a dozen states, so a few zipcode listings are easy to handle.

There's nothing unusual about the report on the number of years as a store owner, but the matter of being a writer for the *Fancy Fish Fancier* might have been derived from a series of questions like this:

Have you written articles for professional journals?

Which? *Fancy Fish Fancier? Bubbles? Fin?*

Have you spoken to professional group meetings?

National? Regional? State? Local?

Have you received professional awards?

Etc...

When this information is fed to the computer for processing by the Dots program, the system will print out:

SUMMARY OF RESULTS

	1	2	3	4	5
1. Topic	27	15	4	82	22
2. Type of Car	16	26	20	69	69
3. Hair	40	40	40	40	40
4. State	20	60	30	55	35
5. Yrs. as Owner	100	30	20	40	10

Personal judgment comes in when the Dot-master selects categories and assigns colors. He may select groups both large and small, trying to develop a satisfying blend of broadly common factors and unusual distinguishing characteristics. The purpose of the effort is to encourage friendly conversation among strangers. So the selection is made with that in mind.

He may select this way:

SUMMARY OF RESULTS

	1	2	3	4	5
1. Topic	27	15	4	82	22
2. Type of Car	16	26	20	69	69
3. Hair	40	40	40	40	40
4. State	20	60	30	55	35
5. Yrs. as Owner	100	30	20	40	10

Now, he tells the computer about his choices.

```
01
ENTER VALUE 1-5
1
ENTER COLOR-MAXIMUM 8 LETTERS
red
02
ENTER VALUE 1-5
4
ENTER COLOR-MAXIMUM 8 LETTERS
white
```

And so on. One group of respondents in each category is now identified by a color. With this information, the computer can assign appropriate dots to each registrant.

```
ID #123 GETS CODED RED
ID #145 GETS CODED BLUE, YELLOW
ID # 98 GETS CODED RED, WHITE
ID # 29 GETS CODED GREEN, ORANGE
..........
..........
..........
```

It is only necessary then to make up the badges themselves with numbers, names and dots.

There are many variables, of course. The number of categoris may be as high as 20 in the present program (though Mark James comments that when the number of dots on the badge approaches 10, things get a bit crowded). The number of possible responses in a given category is variable. The number of badges to be processed is variable, with the maximum set at 300.

Obviously, an automatic system could easily print out finished badges, calling only for attachment of colored stickers or some work with marking pens. The Lemonade Entrepreneur will think of a dozen special features and services he can add to this activity to serve his customers well.

A nice touch: *Everybody* gets at least one dot. After the program has sorted everbody out, it checks the original list to see if anyone has been left out. The leftover folks are then all assigned a dot color of their own. What they have in common, then, is that they have nothing *else* in common (that anybody knows about) with other colored dot people. That's a valid distinction, well worth a conversation with a similarly dotted stranger.

After your long trip and an attack of cold feet, you may take comfort in the fact that dots break the ice.

Comex is the leading U.S. Center for the Study and Development of Gaming and Simulation. PERSONAL COMPUTING will be publishing more of the Center's work as it becomes available. Address inquiries to: Comex, Davidson Conference Center, Univ. of Southern California, Los Angeles, CA 90007.

LEMONADE COMPUTER SERVICE

Frozen conference attendees, anxious to melt away their timid feelings, can take relief in connecting the Dots. And the Lemonade Entrepreneur, armed with his trusty computer, stands to make a buck.

```
 KAE01.DOTS.FORT
C
C        DOTS PROGRAM DEVELOPED BY MARK JAMES FOR THE COMEX PROJECT
C        OF THE UNIVERSITY OF SOUTHERN CALIFORNIA (12-76).
C        THE PURPOSE OF DOTS IS TO FIND PEOPLE OF SIMILAR
C        CHARACTERISTICS ATTENDING CONFERENCE.
C
C
      DIMENSION MAT(20,5),CAT(20,5)
      DIMENSION ID(300),IRES(300,20)
      DIMENSION COL(10,2),JCAT(10),JVAL(10),ZL(20)
      DIMENSION XCOL(2)
C
C
C        MAT=FREQUENCY MATRIX OF RESPONSES
C
C
C
C        ID=MAXIMUM OF 300 PARTICIPANTS
C
C        COL=MAXIMUM OF 10 COLORS FOR GROUPINGS
C
C        JCAT=CATEGORIES TO BE SELECTED
C
C        FVAL=VALUES TO BE SELECTED
C
C        ZL=PRINT LINE OF RESULTS
C
C
C
      DATA MAT/100*0/,BLNK/'     '/
      ICAT=0
C
C
C        ENTER CATEGORIES
C
C
      WRITE(6,100)
100   FORMAT(' ENTER A CATEGORY TITLE-MAXIMUM 20 LETTERS-ENTER BLANK LIN
     1E WHEN FINISHED')
1     ICAT=ICAT+1
      IF(ICAT .GT. 20) GO TO 2
      WRITE(6,102) ICAT
102   FORMAT(' CATEGORY',I3)
      READ(1,101)(CAT(ICAT,J),J=1,5)
      IF(CAT(ICAT,1) .NE. BLNK) GO TO 1
      ICAT=ICAT-1
C
C
C        ENTER INDIVIDUAL RESPONSES
C
C
2     WRITE(6,200)
200   FORMAT(' ENTER 3 DIGIT ID # PLUS RESPONSES FOR EACH CATEGORY------
     1 MAXIMUM VALUE PER RESPONSE IS 5'/
     2' ENTER BLANK LINE WHEN FINISHED')
      NUMB=0
3     WRITE(6,300)
300   FORMAT(' ID #')
      NUMB=NUMB+1
      IF(NUMB .GT. 300) GO TO 4
      READ(1,301) ID(NUMB),(IRES(NUMB,J),J=1,ICAT)
```

```
      IF(ID(NUMB) .EQ. 0) GO TO 5
      GO TO 3
101   FORMAT(5A4)
301   FORMAT(I3,20I1)
4     WRITE(6,400)
400   FORMAT(' MAXIMUM OF 300 ENTRIES ALLOWED---')
5     NUMB=NUMB-1
C
C
C        PLACE RESPONSES IN MATRIX AND TABULATE FREQUENCY OF RESULTS
C
C
      DO 10 I=1,NUMB
      DO 20 J=1,ICAT
      IF(IRES(I,J) .GT. 0.AND. IRES(I,J) .LT. 6)
     1  MAT(J,IRES(I,J))=MAT(J,IRES(I,J)) +1
20    CONTINUE
10    CONTINUE
      WRITE(6,500) (I,I=1,5)
500   FORMAT(////28X,'SUMMARY OF RESULTS'/23X,5I5/)
      DO 30 I=1,ICAT
30    WRITE(6,600) I,(CAT(I,J),J=1,5),(MAT(I,K),K=1,5)
600   FORMAT(' ',I2,1X,5A4,5I5)
      WRITE(6,601)
601   FORMAT(/////)
      ICOL=0
7     ICOL=ICOL+1
17    WRITE(6,700)
C
C
C        ORGANIZERS SELECT GROUPINGS OF CATEGORIES AND RESPONSES
C
C
700   FORMAT(' ENTER CATEGORY # (I2)-BLANK LINE TO END LOOP')
      IF(ICOL .GT. 10) GO TO 6
      READ(1,701) JCAT(ICOL)
      IF(JCAT(ICOL) .GT. ICAT) GO TO 17
701   FORMAT(I2)
      IF(JCAT(ICOL))6,6,8
8     WRITE(6,702)
702   FORMAT(' ENTER VALUE 1 TO 5-(I1)')
      READ(1,703) JVAL(ICOL)
703   FORMAT(I1)
      IF(JVAL(ICOL) .GT. 0 .AND. JVAL(ICOL) .LT. 6) GO TO 9
      GO TO 8
9     WRITE(6,704)
704   FORMAT(' ENTER COLOR-MAXIMUM 8 LETTERS')
      READ(1,705) (COL(ICOL,J),J=1,2)
705   FORMAT(2A4)
      IF(COL(ICOL,1) .EQ. BLNK) GO TO 9
      GO TO 7
6     WRITE(6,801)
801   FORMAT(' ENTER A COLOR FOR THOSE PEOPLE WHO DO NOT FIT ANY ',
     1 'CATEGORY-MAX. 8 LETTERS')
      READ(1,705) XCOL
      WRITE(6,601)
      ICOL=ICOL-1
C
C
C        PRINT OUT RESULTS
C
C
      DO 50 I=1,NUMB
      IX=0
      DO 55 K=1,20
55    ZL(K)=BLNK
      DO 60 J=1,ICOL
      IF(IRES(I,JCAT(J)) .NE. JVAL(J)) GO TO 60
      DO 65 K=1,2
      IX=IX+1
65    ZL(IX)=COL(J,K)
60    CONTINUE
      IF(IX .EQ. 0) GO TO 51
53    WRITE(6,800) ID(I),(ZL(K),K=1,IX)
800   FORMAT(' ID #',I4,2X,'GETS CODED',1X,10(2A4,',',1X))
      GO TO 50
51    DO 52 K=1,2
      IX=IX+1
52    ZL(IX)=XCOL(K)
      GO TO 53
50    CONTINUE
      WRITE(6,601)
      STOP
      END
$ENTRY
READY
```

LEMONADE COMPUTER SERVICE

RECURSIVE BUDGETING

* * *

by O.E. Dial

Everybody talks about using the personal computer for controlling personal budgets — but Gene Dial has done something about it. If you have been calculating your expenditures by drawing your creditors' names from a hat, you'll find this sensible, straightforward program a lifesaver. . . and if you're a Lemonade Entrepreneur at heart, you may find yourself earning a few dollars by helping customers get a grip on their budgets.

Budgeting can be difficult: behaviorally, where several persons must reach agreement as to its allocations; and mechanically, where many accounts must be balanced within an upper limit. Both these features of budgeting are much in evidence when a family, business or city council undertake the task. Since both sources of difficulty can be eased substantially where the task is computer-assisted, the economies of the microcomputer make it feasible to provide such assistance. The program in this article is intended to help you provide assistance to yourself or others.

Behaviorally, each participant must be allowed to value his own priorities and have these reflected in the budget, at least at the outset. Confrontation with others who may disagree is postponed. More importantly, the arena of conflict shrinks with each iteration of the budgeting process. First, this shrinkage occurs partly because the budget for each account is reduced pro-rata upon the completion of each iteration so as to reconcile with the budget ceiling. Thus, the budgeters find themselves differing with the computer rather than a personality. Second, as agreement is reached with a satisfactory level of confidence as to the level of a particular account, the account is frozen at that level and removed from further review or adjustment. Each iteration then yields fewer and fewer accounts upon which to disagree (or agree).

Mechanically, the pencil, pad of paper, hours of arithmetic (each adjustment requires a compensating adjustment in one or more other accounts) and the inevitable errors are side-stepped. What had been periodic trauma second only to sex as the cause of divorce is now reduced to a pleasant, entertaining experience duly celebrated with appropriate libation.

This article will provide an overview of the program and its use: the accompanying pages provide an illustrative walk-through. Much of the program is conversational with the computer, and the remainder consists of output reports.

```
==============================================================

                 RECURSIVE BUDGETING MODEL

                           * * *

==============================================================

                 SPENDABLE INCOME SUMMARY

--------------------------------------------------------------
   ACCOUNT                              PERIOD         ANNUAL
--------------------------------------------------------------
TOTAL INCOME                            $1,605        $19,260

   PAYCHECK DEDUCTIONS        -$367

   FIXED EXPENSES             -$554      -$921       -$11,052
                                       -------       --------
SPENDABLE INCOME                          $684         $8,208
                                       =======       ========

==============================================================
```

1. A summary shows the balance available for spendable income.

```
PRESS 'Y' TO CONTINUE

? Y

OK,NOW FOR THE FIRST ROUND OF VARIABLE EXPENSE. DON'T
PINCH YOURSELF IN YOUR ESTIMATES (WITHIN REASON). LET THE
COMPUTER HELP YOU REFINE YOUR BUDGET LATER ON.

FOOD/BEVERAGES $? 180
CLOTHING $? 25
DRY CLEANING $? 15
BARBER/BEAUTY $? 20
HOME MAINT $? 15
HOME HEAT'G FUEL $? 50
WATER $? 45
ELECTRICITY $? 25
TELEPHONE $? 35
GAS/OIL $? 42
AUTO MAINT $? 25
FARES/TOLLS/PARKING $? 20
DENTIST $? 15
PHYSICIAN $? 10
DRUGS/SUNDRIES $? 12
SCHOOL EXPENSE $? 35
FAMILY ALLOWANCE $? 50
CLUBS/LODGES $? 50
THEATER/SPORTS $? 10
RESTAURANTS $? 20
OTHER ENT'MENT $? 10
MAG'S/BOOKS/PAPERS $? 8
SITTERS $? 15
CHILD CARE $? 35
VACATION SAVINGS $? 50
OTHER SAVINGS $? 75
CONTRIBUTIONS $? 15
OTHER EXPENSES $? 14

YOUR BUDGET FOR THE FIRST ROUND TOTALLED $ 921 . THIS
COMPARES TO SPENDABLE INCOME OF $ 684 . WE HAVE
PRORATED THE DIFFERENCE, $-237 , OVER ALL VARIABLE EXPENSE
ACCOUNTS.

PRESS 'Y' TO CONTINUE

? Y
```

INTERIM SUMMARY OF VARIABLE EXPENSES: NO. 1

ACCOUNTS	NUMBER	PERCENT	AMOUNT	PERCENT
VARIABLE EXPENSES	12	48.0	$261	16.3
FROZEN EXPENSES	13	52.0	$423	26.4
TOTALS	25		$684	

INTERIM STATUS OF VARIABLE EXPENSES NO. 1

ACCOUNT	VARIABLE	FROZEN	PERCENT
FOOD/BEVERAGES		$150	9.3
CLOTHING	$9		0.6
DRY CLEANING		$15	0.9
BARBER/BEAUTY	$8		0.5
HOME MAINT	$9		0.6
HOME HEAT'G FUEL		$36	2.2
WATER		$32	2.0
ELECTRICITY		$25	1.6
TELEPHONE		$24	1.5
GAS/OIL		$40	2.5
AUTO MAINT		$15	0.9
FARES/TOLLS/PARKING		$16	1.0
DENTIST	$5		0.3
PHYSICIAN	$7		0.4
DRUGS/SUNDRIES	$6		0.4
SCHOOL EXPENSE	$19		1.2
FAMILY ALLOWANCE	$37		2.3
CLUBS/LODGES		$35	2.2
THEATER/SPORTS	$8		0.5
RESTAURANTS	$11		0.7
SITTERS		$10	0.6
CHILD CARE		$20	1.2
VACATION SAVINGS	$31		1.9
OTHER SAVINGS	$48		3.0
CONTRIBUTIONS		$5	0.3

To print the conversation and the reports would require yards of paper, only a portion of which would be needed for later reference. For this reason, both the video (for conversation) and the printer (for reports) are employed. The movement from one to the other is of course fully automated in the program. Furthermore, where there is a danger of the program racing ahead of the user while in the video mode, PRESS 'Y' TO CONTINUE statements are liberally employed throughout.

The model consists of three parts, of which the second, the budget adjustment routine, is iterative. The first part requires inputting the budgeting period with income deductions from income and fixed expense data. The budgeting period normally would correspond to the pay period of the user. In the final report, the budget is automatically extended to an annual basis.

Wages or salary and other income constitute the budget limit. From this total payroll deductions and fixed expenses are subtracted. The program lists the usual deductions and fixed expenses to which the user responds with appropriate inputs. "Other" is of course appended to each listing as a catch-all for accounts the program does not anticipate. When these inputs are provided the program shifts to printer and outputs a summary report that calculates spendable income, the budget ceiling within which the balance of the program must be constrained.

The program then proceeds to variable expenses. Here, some 28 typical expense categories are listed, again concluding with a catch-all. The user is encouraged to enter the desired budget level for each account without being unduly concerned with the budget ceiling. What is being established is the relative priority of each budget account. When the 28th entry is completed, the program calculates the total and reduces (or increases) the budget for each account on a pro-rata basis weighted according to the relative account level inputted for each. For

2. The user is advised against restraint in inputting the initial budget levels for each account. After inputting levels for each account, the user is shown the total of his inputs for variable expense accounts and the amount by which he has exceeded his budget.

3. Interim reports are published upon the completion of each iteration. The first provides a summary and the second, greater detail with respect to the number and budget allowance for each account in the variable and frozen lists. The reports are numbered for convenience in later reference.

example, assuming two accounts, one having twice the level of the other, and assuming further that an overall budget reduction must be made, the larger account would suffer twice the reduction of the smaller account. The budget is now tending toward reality, but has not yet arrived.

The second part of the program, the iterative part is now entered. For each category, the computer displays the name of the account and its computer-adjusted budget level. It then asks if the user desires to revise the account level. Should the user reply affirmatively, the computer asks for the revised amount. If the revised amount exceeds the available balance in spendable income (the total of the remaining variable expense accounts which have not yet been frozen), the user will be advised of the amount of the coverage and asked to input a new revision.

This dialog is followed by a remaining question for the same account. Does the user desire to freeze the account? He is encouraged to go slowly in deciding to freeze an account. To freeze means that the account is removed from the variable expense list and placed in the frozen list. The frozen list is not subject to further review for purposes of adjustment. For that matter, accounts for which a 0 has been inputted are not again presented for review. Where the user entertains any doubt at all about the use of an account, he should enter at least some small value so as to permit a future decision as to its use.

Each time a revision is entered, the budget ceiling for all variable expense accounts is recalculated. This procedure ensures that the total budget remains within the budgetary ceiling at all times, at least upon completion of the first part of the program. Those accounts which have been removed to the frozen matrix are of course unaffected by this recalculation.

Upon the conclusion of each iteration of this program part, a budget summary is printed as an assist in the next following iteration. The summary reports the number of accounts in the frozen and variable lists, and the percentage of each with respect to the total number of accounts; and the total amounts for each class, and the percentages of each with respect to total in-

* THE JOHN Q. DOE FAMILY BUDGET FOR 1977 *

ACCOUNT	INCOME		EXPENSE		PERCENT TOTAL
	PERIOD	ANNUAL	PERIOD	ANNUAL	INCOME
INCOME	$1,605	$19,260			100.0

SALARY/WAGES	$1,560	$18,720			97.2
OTHER INCOME	$45	$540			2.8
PAYCHECK DEDUCT'S			$367	$4,404	22.9

FED INC TAX			$225	$2,700	14.0
STATE & LOCAL TAX			$32	$384	2.0
SOCIAL SECURITY			$65	$780	4.0
HEALTH INS			$35	$420	2.2
CONTRIBUTIONS			$10	$120	0.6
FIXED EXPENSES			$554	$6,648	34.5

RENT/MORTGAGE			$275	$3,300	17.1
LIFE INS			$35	$420	2.2
HOUSE INS			$17	$204	1.1
AUTO INS			$21	$252	1.3
CAR PAYMENTS			$125	$1,500	7.8
LOAN PAYMENTS			$75	$900	4.7
TRASH REMOVAL			$6	$72	0.4
VARIABLE EXPENSE			$684	$8,208	42.6

FOOD/BEVERAGES			$159	$1,908	9.9
CLOTHING			$10	$120	0.6
DRY CLEANING			$15	$180	0.9
BARBER/BEAUTY			$12	$144	0.7
HOME MAINT			$10	$120	0.6
HOME HEAT'G FUEL			$36	$432	2.2
WATER			$32	$384	2.0
ELECTRICITY			$25	$300	1.6
TELEPHONE			$24	$288	1.5
GAS/OIL			$40	$480	2.5
AUTO MAINT			$15	$180	0.9
FARES/TOLLS/PARKING			$16	$192	1.0
DENTIST			$10	$120	0.6
PHYSICIAN			$10	$120	0.6
DRUGS/SUNDRIES			$10	$120	0.6
SCHOOL EXPENSE			$20	$240	1.2
FAMILY ALLOWANCE			$35	$420	2.2
CLUBS/LODGES			$35	$420	2.2
THEATER/SPORTS			$10	$120	0.6
RESTAURANTS			$12	$144	0.7
SITTERS			$10	$120	0.6
CHILD CARE			$20	$240	1.2
VACATION SAVINGS			$35	$420	2.2
OTHER SAVINGS			$45	$540	2.8
CONTRIBUTIONS			$5	$60	0.3

4. The first of the two-page report. The "Percent of Total Income" column is particularly useful for year-to-year comparisons.

TABLE OF VARIABLES

A	TEMPORARY VALUES
A$	TEMPORARY NAMES
B	LENGTH OF NAME COUNTER
B$	TEMPORARY NAME
D(J)	VALUES OF INCOME-DEDUCTIBLES-FIXED EXPENSES
DF	TOTAL OF DEDUCTIBLES AND FIXED EXPENSES (TD+TF)
E$(J)	NAMES OF VARIABLE EXPENSE ACCOUNTS
F(J)	VALUES OF FROZEN EXPENSE ACCOUNTS
F1	FROZEN EXPENSE ACCOUNT COUNTER
FT	TOTAL OF FROZEN EXPENSES
J	PRIMARY LOOP INDEX
K	SECONDARY LOOP INDEX
L	TERTIARY LOOP INDEX--FOR SUBROUTINES
N	INPUT RESPONSE--NO
P	BUDGETING PERIOD
Q	NUMBER OF VARIABLE EXPENSE ACCOUNTS
R	TEMPORARY COUNTER
R0	ITERATION COUNTER
S	WIDTH OF PRINTED TABLES; LENGTH OF BAR CHART BARS
SI	SPENDABLE INCOME (TI-DF)
TD	TOTAL DEDUCTIONS
TF	TOTAL FIXED EXPENSES
TI	TOTAL INCOME
U$	PRINT USING FORMAT FOR PERCENTAGES TO ONE DECIMAL PLACE
V$	PRINT USING FORMAT FOR INTEGER PERCENTAGES AND COUNTS
V(J)	VALUES OF VARIABLE EXPENSE ACCOUNTS
V1	VARIABLE EXPENSE ACCOUNT COUNTER
VT	TOTAL OF VARIABLE EXPENSES
W$	PRINT USING FORMAT FOR DOLLAR AMOUNTS
Y	INPUT RESPONSE--YES

5. A table of the variables used in the program. This listing should help clarify the program code.

```
                    * EXPENSE BAR CHART *
                   PERCENT OF TOTAL INCOME

 -----------------------------------------------------------------

                       - - - - 5 - - - - 10 - - - - 15 - -

FED INC.TAX           ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
STATE & LOCAL TAX     ]]]]]
SOCIAL SECURITY       ]]]]]]]]]]]]
HEALTH INS            ]]]]]]
CONTRIBUTIONS         ]]
RENT/MORTGAGE         ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
LIFE INS              ]]]]]]
HOUSE INS             ]]]
AUTO INS              ]]]
CAR PAYMENTS          ]]]]]]]]]]]]]]]]]]]]]]]]
LOAN PAYMENTS         ]]]]]]]]]]]]]]]
TRASH REMOVAL         ]]
FOOD/BEVERAGES        ]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]]
CLOTHING              ]]
DRY CLEANING          ]]
BARBER/BEAUTY         ]]
HOME MAINT            ]]
HOME HEAT'G FUEL      ]]]]]]
WATER                 ]]]]]
ELECTRICITY           ]]]]
TELEPHONE             ]]]]
GAS/OIL               ]]]]]]]
AUTO MAINT            ]]
FARES/TOLLS/PARKING   ]]
DENTIST               ]]
PHYSICIAN             ]]
DRUGS/SUNDRIES        ]]
SCHOOL EXPENSE        ]]]
FAMILY ALLOWANCE      ]]]]]]
CLUBS/LODGES          ]]]]]]
THEATER/SPORTS        ]]
RESTAURANTS           ]]
SITTERS               ]]
CHILD CARE            ]]]
VACATION SAVINGS      ]]]]]]
OTHER SAVINGS         ]]]]]]]]
CONTRIBUTIONS         ]

                       - - - - 5 - - - - 10 - - - - 15 - -
```

```
                 RECURSIVE BUDGETING CODE LIST
              =================================

LIST

10 Q=27:V$="###.#":W$="$$#####,":U$="###"
20 DIMD(18),E$(Q),V(Q),F(Q)
30 PRINTTAB(19)"RECURSIVE BUDGETING MODEL":PRINT:PRINTTAB(28)"* * *"
40 DATASALARY/WAGES,OTHER INCOME,FED INC TAX,STATE & LOCAL TAX
41 DATASOCIAL SECURITY,UNEMPLOYMENT INS,HEALTH INS
42 DATALIFE INS,CONTRIBUTIONS,OTHER DEDUCTIONS
43 DATARENT/MORTGAGE,LIFE INS,HEALTH INS,HOUSE INS
44 DATAAUTO INS,CAR PAYMENTS,LOAN PAYMENTS,TRASH REMOVAL
45 DATAOTHER FIXED EXP
50 DATAFOOD/BEVERAGES,CLOTHING,DRY CLEANING,BARBER/BEAUTY
51 DATAHOME MAINT,HOME HEAT'G FUEL,WATER,ELECTRICITY,TELEPHONE
52 DATAGAS/OIL,AUTO MAINT,FARES/TOLLS/PARKING,DENTIST
53 DATAPHYSICIAN,DRUGS/SUNDRIES,SCHOOL EXPENSE,FAMILY ALLOWANCE
54 DATACLUBS/LODGES,THEATER/SPORTS,RESTAURANTS
55 DATAOTHER ENT'MENT,MAG'S/BOOKS/PAPERS,SITTERS,CHILD CARE
56 DATAVACATION SAVINGS,OTHER SAVINGS,CONTRIBUTIONS,OTHER EXPENSES
70 PRINT:PRINT"SELECT YOUR BUDGETING PERIOD BY NUMBER. LATER ON IT WILL"
72 PRINT"BE EXTENDED TO ONE YEAR.":PRINT
80 PRINTTAB(3)"1-WEEKLY"TAB(15)"2-BIWEEKLY"TAB(30)"3-SEMIMONTHLY";
82 PRINTTAB(45)"4-MONTHLY":PRINT
90 INPUTP:IFP>4THENPRINT"TRY AGAIN":GOTO70
100 IFP=1THENP=52ELSEIFP=2THENP=26ELSEIFP=3THENP=24ELSEIFP=4THENP=12
110 PRINT:PRINT"ALRIGHT,FIRST LET'S LOOK AT INCOME FOR THE PERIOD.":PRINT
120 READA$:PRINTA$;" $";:INPUTD(0):READA$:PRINTA$;" $";:INPUTD(1)
122 TI=D(0)+D(1):PRINT
140 PRINT:PRINT"OK,NOW LET'S LOOK AT PAYCHECK DEDUCTIONS.":PRINT
150 FORJ=2TO9:READA$:PRINTA$;:INPUT" $";D(J):TD=TD+D(J):NEXTJ:PRINT
160 PRINT"OK,NOW LET'S LOOK AT FIXED EXPENSES.":PRINT
170 FORJ=10TO18:READA$:PRINTA$;:INPUT" $";D(J):TF=TF+D(J):NEXTJ
180 DF=TD+TF:SI=TI-DF:S=64
185 PRINT:PRINT"OK,AT THIS TIME OUR TABLE LOOKS LIKE THIS:":PRINT
190 OUT18,3:OUT18,17
195 CONSOLE18:PRINT
200 GOSUB1205:PRINT:PRINTTAB(19);
201 PRINT"RECURSIVE BUDGETING MODEL":PRINT:PRINTTAB(27)"* * *":PRINT
202 GOSUB1205
205 PRINT:PRINTTAB(20)"SPENDABLE INCOME SUMMARY":PRINT
210 GOSUB1200:PRINTTAB(3)"ACCOUNT"TAB(42)"PERIOD"TAB(57)"ANNUAL"
215 GOSUB1200:PRINT"TOTAL INCOME" TAB(40);:PRINTUSINGW$;TI;:PRINTTAB(55)
220 PRINTUSINGW$;TI*P:PRINT:PRINTTAB(3)"PAYCHECK DEDUCTIONS"TAB(25);
225 PRINTUSINGW$;TD*(-1):PRINT
230 PRINTTAB(3)"FIXED EXPENSES"TAB(25);:PRINTUSINGW$;TF*(-1);
240 PRINTTAB(40);:PRINTUSINGW$;DF*(-1);:PRINTTAB(55);
244 PRINTUSINGW$;DF*(-1)*P
250 PRINTTAB(41)"--------"TAB(55)"---------":PRINT"SPENDABLE INCOME";
```

Continued on next page.

come. Then follows a more detailed report that gives the level of each account in each class. The user is then advised of the percentage of his task completed, i.e., the percentage of variable expense accounts transferred to the frozen matrix.

The user will find that with each iteration he becomes increasingly confident of the budget levels set for particular accounts. And so it is, that with each iteration more and more accounts are transferred to the frozen matrix. Eventually, all have found their way. At this point the program enters its last part, the final report phase. This report details each income and expense account under appropriate heads and extends each to an annual basis. A final column reports each income and expense account level as a percentage of total income. This report is followed by a bar chart with a dimensioned bar for each deduction, fixed expense and variable expense account.

The program has been tested with neighbors and colleagues. Their comments not only verified its utility but the behavioral aspects discussed in this article. Since it might be interesting to evaluate the correlation between the numbers of participants in the budgeting process and the number of iterations required for its resolution, notes are being kept for each serious use of the program.

The program language is Mits Extended (12K) Basic. The listing requires about 10K words of memory when passive. This capacity is doubled, however, in the active mode. Additionally, to be on the safe side, some 3.5K words of string space should be reserved. The program listing can probably be reduced by one-third with the removal of various amenities and aesthetics, i.e., extensive use of literals, PRINT USING's, columnar markings, underscoring and so on. On the other hand, it can profitably be enlarged to permit such desired features as (1) the recall and reconsideration of frozen accounts, (2) the correction of entries in the income and expense accounts, (3) the inputting of additional account titles and (4) the automated reconciliation of annual costs to the budget period so as to re-

6. The final output document lets you recognize significant expense accounts at a glance.

7. Here's the program listing. The programming language is Mits Extended (12K) Basic. This listing requires about 10K words of memory when passive and about twice as much in the active mode. To be on the safe side reserve an extra 3-5K words for strings.

```
255 PRINTTAB(40);:PRINTUSINGW$;SI;:PRINTTAB(55);:PRINTUSINGW$;SI*P
260 PRINTTAB(41)"======="TAB(55)"========":PRINT:
270 GOSUB1205:PRINT: PRINT
280 OUT18,3:OUT18,17
290 CONSOLE16
300 GOSUB1210:RO=0
310 PRINT:PRINT "OK,NOW FOR THE FIRST ROUND OF VARIABLE EXPENSE. DON'T"
312 PRINT"PINCH YOURSELF IN YOUR ESTIMATES (WITHIN REASON). LET THE"
314 PRINT"COMPUTER HELP YOU REFINE YOUR BUDGET LATER ON.":PRINT
320 FORJ=0TOQ:READE$(J):PRINTE$(J);:INPUT" $";V(J):VT=VT+V(J):NEXTJ
330 RESTORE:PRINT:PRINT
340 PRINT"YOUR BUDGET FOR THE FIRST ROUND TOTALLED $"VT". THIS"
342 PRINT"COMPARES TO SPENDABLE INCOME OF $"SI". WE HAVE"
344 PRINT"PRORATED THE DIFFERENCE, $"SI-VT", OVER ALL VARIABLE EXPENSE"
346 PRINT"ACCOUNTS.":PRINT
350 GOSUB1210:FORJ=0TOQ:V(J)=INT(V(J)/VT*SI):NEXTJ:VT=SI:PRINT
360 PRINT"NOW WE BEGIN THE BUDGET REFINEMENT PHASE. MAKE AS MANY"
362 PRINT"PASSES AS YOU LIKE. AS YOU REVIEW EACH ACCOUNT,DECIDE"
364 PRINT"WHETHER TO FREEZE IT OR TO LEAVE IT FOR ANOTHER PASS.":PRINT
366 PRINT"HINT: DON'T BE IN A HURRY TO FREEZE AN ACCOUNT.":PRINT
368 PRINT"YOUR TASK IS FINISHED WHEN ALL ACCOUNTS ARE FROZEN.":PRINT
370 GOSUB1210
400 FORJ=0TOQ:PRINT:IFV(J)=0THEN531
410 PRINTE$(J);" $";V(J):INPUT"CHANGE ('Y' OR 'N')";A$
412 IFA$="N"THEN430ELSEIFA$="Y"THEN420ELSEIFA$<>"Y"THEN410
415 GOTO410
420 INPUT"REVISED AMOUNT $";A:IFA<V(J)THEN430
425 GOSUB1230
430 INPUT"FREEZE ('Y' OR 'N')";B$:PRINT
440 IFA$="Y"ANDB$="Y"THEN480
442 IFA$="Y"ANDB$="N"THEN510
450 IFA$="N"ANDB$="N"THEN531
460 VT=VT-V(J):FT=FT+V(J):SWAPF(J),V(J):V(J)=0:PRINT
470 PRINT"OK,THE ACCOUNT HAS BEEN FROZEN AT $"F(J)".":GOTO531
480 VT=VT-A:FT=FT+A:SWAPA,F(J):V(J)=0
490 FORK=0TOQ:V(K)=INT(V(K)*(SI-FT)/VT):NEXTK:VT=SI-FT
500 PRINT"OK,THE ACCOUNT HAS BEEN FROZEN AT $"F(J)",AND THE DIFFERENCE"
502 PRINT"PRORATED OVER REMAINING ACCOUNTS.":PRINT:GOTO531
510 VT=VT-V(J)+A:V(J)=A
512 FORK=0TOQ:V(K)=INT(V(K)*(SI-FT)/VT):NEXTK:VT=SI-FT
520 PRINT"OK,THE VALUE HAS BEEN CHANGED TO $"A", AND THE DIFFERENCE"
522 PRINT"PRORATED OVER REMAINING ACCOUNTS.":PRINT
531 NEXTJ:PRINT:R=0:FORJ=0TOQ:IFV(J)=0THENR=R+1:NEXTJ
532 IFR<=QTHEN540
534 FORJ=0TOQ:F(J)=INT(F(J)/FT*SI):VT=0:FT=SI:GOTO800
540 PRINT"NOW LET'S RECAP VARIABLE EXPENSES BEFORE GOING ON TO THE"
542 PRINT"NEXT ROUND.":PRINT:RO=RO+1:V1=0:F1=0
543 FORJ=0TOQ:V(J)=INT(V(J)/VT*(SI-FT)):NEXTJ:VT=SI-FT
546 OUT18,3:OUT18,17
548 CONSOLE18:PRINT:PRINT
550 PRINTTAB(7)"INTERIM SUMMARY OF VARIABLE EXPENSES";
552 PRINT":   NO."RO
560 GOSUB1200:PRINTTAB(3)"ACCOUNTS"TAB(28)"NUMBER"TAB(37)"PERCENT";
562 PRINTTAB(48)"AMOUNT"TAB(56)"PERCENT"
570 GOSUB1200:FORJ=0TOQ:IFV(J)+F(J)=0THEN590
580 IFV(J)THENV1=V1+1ELSEF1=F1+1
590 NEXTJ
600 PRINT"VARIABLE EXPENSES"TAB(29);:PRINTUSINGU$;V1;:PRINTTAB(39);
602 PRINTUSINGV$;V1/(V1+F1)*100;:PRINTTAB(46);:PRINTUSINGW$;VT;
604 PRINTTAB(58);:PRINTUSINGV$;VT/TI*100:PRINT
610 PRINT"FROZEN EXPENSES"TAB(29);:PRINTUSINGU$;F1;:PRINTTAB(39);
612 PRINTUSINGV$;F1/(V1+F1)*100;:PRINTTAB(46);
614 PRINTUSINGW$;FT;:PRINTTAB(58);:PRINTUSINGV$;FT/TI*100
616 PRINTTAB(28)"--------"TAB(48)"---------"
620 PRINTTAB(3)"TOTALS"TAB(29);:PRINTUSINGU$;F1+V1;:PRINTTAB(46);
622 PRINTUSINGW$;VT+FT:PRINTTAB(28)"======="TAB(48)"========":PRINT
630 GOSUB1205:PRINT:PRINT
650 GOSUB655:GOTO680
655 PRINTTAB(8)"INTERIM STATUS OF VARIABLE EXPENSES  NO."RO
660 GOSUB1200:PRINTTAB(3)"ACCOUNT"TAB(30)"VARIABLE"TAB(43)"FROZEN";
662 PRINTTAB(54)"PERCENT"
670 GOSUB1200:PRINT
680 FORJ=0TOQ:IFV(J)+F(J)=0THEN740ELSEPRINTE$(J);:R2=R2+1
690 IFF(J)THEN700ELSEPRINTTAB(29):PRINTUSINGW$;V(J);:PRINTTAB(55);
695 PRINTUSINGV$;V(J)/TI*100:GOTO720
700 PRINTTAB(40);:PRINTUSINGW$;F(J);:PRINTTAB(55);
710 PRINTUSINGV$;F(J)/TI*100
720 IFINT(R2/5)=R2/5THENPRINT
740 NEXTJ:PRINT:R2=0
750 GOSUB1205:PRINT
752 OUT18,3:OUT18,17
754 CONSOLE16
770 PRINT:PRINT "NOW WE MAKE ANOTHER PASS THROUGH VARIABLE EXPENSES.":PRINT
780 GOTO370
800 PRINT"PLEASE INPUT THE REQUESTED INFORMATION BEFORE WE PROVIDE A"
802 PRINT"FINAL DETAIL OF YOUR BUDGET.":PRINT
804 PRINTTAB(18)"* USE NO COMMAS NOR COLONS *":PRINT
810 INPUT"NAME";A$:INPUT"BUDGET YEAR";A
812 B=LEN(A$):B=16-B/2
820 OUT18,3:OUT18,17
822 CONSOLE18:PRINT:PRINT
830 GOSUB1220
840 GOSUB1205:PRINT
850 PRINTTAB(B)"* THE "A$" FAMILY BUDGET FOR"A"*":PRINT
860 GOSUB1205:PRINT
870 PRINTTAB(6)"ACCOUNT"TAB(20)"|"TAB(26)"INCOME"TAB(38)"|";
872 PRINTTAB(44)"EXPENSE"TAB(56)"|PERCENT"
880 PRINTTAB(20)"|-----------------|-----------------| TOTAL"
890 PRINTTAB(20)"| PERIOD | ANNUAL | PERIOD | ANNUAL |INCOME"
900 PRINT"-------------------|--------|--------|--------|--------|-------"
910 X$="|$$######,|$$######,|        |        | ###.#"
912 Y$="|        |        |$$######,|$$######,| ###.#"
914 GOTO930
920 PRINT"|        |        |        |        |":RETURN
930 PRINT"INCOME"TAB(20);:PRINTUSINGX$;TI;TI*P;TI/TI*100
932 PRINT"------"TAB(20);
934 GOSUB920
936 READA$:PRINTA$TAB(20);:PRINTUSINGX$;D(0);D(0)*P;D(0)/TI*100
938 READA$:PRINTA$TAB(20);:PRINTUSINGX$;D(1);D(1)*P;D(1)/TI*100
940 PRINTTAB(20);
942 GOSUB920
950 PRINT"PAYCHECK DEDUCT'S"TAB(20);
951 PRINTUSINGY$;TD;TD*P;TD/TI*100
952 PRINT"-----------------"TAB(20);
954 GOSUB920
```

```
956 FORJ=2TO9:READA$:IFD(J)=0THEN960ELSEPRINTA$TAB(20);
958 PRINTUSINGY$;D(J);D(J)*P;D(J)/TI*100
960 NEXTJ:PRINTTAB(20);
974 GOSUB920
976 PRINT"FIXED EXPENSES"TAB(20);
977 PRINTUSINGY$;TF;TF*P;TF/TI*100
978 PRINT"---------------"TAB(20);
979 GOSUB920
980 FORJ=10TO18:READA$:IFD(J)=0THEN986ELSEPRINTA$TAB(20);
982 PRINTUSINGY$;D(J);D(J)*P;D(J)/TI*100
986 NEXTJ:PRINTTAB(20);
988 GOSUB920
990 PRINT"VARIABLE EXPENSE"TAB(20);
992 PRINTUSINGY$;FT;FT*P;FT/TI*100
994 PRINT"-----------------"TAB(20);
996 GOSUB920
998 FORJ=0TOQ:IFF(J)=0THEN1005ELSEPRINTE$(J)TAB(20);
1000 PRINTUSINGY$;F(J);F(J)*P;F(J)/TI*100
1005 NEXTJ:PRINT
1010 GOSUB1205:PRINT
1020 GOSUB1205:PRINT:GOTO1300
1200 FORL=1TOS:PRINT"-";:NEXTL:PRINT:RETURN
1205 FORL=1TOS:PRINT"=";:NEXTL:PRINT:RETURN
1210 PRINT"PRESS 'Y' TO CONTINUE":PRINT:INPUTA$:PRINT:RETURN
1220 FORL=1TO10:PRINT;:NEXTL:RETURN
1230 IFA<=V(J)THEN1260
1240 IFA<=VTTHEN1260
1250 PRINT"* YOU HAVE EXCEEDED YOUR BALANCE OF SPENDABLE INCOME BY $"A-SI;
1252 PRINT" *":INPUT"REVISED AMOUNT $";A:PRINT
1260 RETURN
1300 RESTORE:READA$:READA$:PRINT:PRINT:S=95
1302 PRINTTAB(21)"* EXPENSE BAR CHART *"
1304 PRINTTAB(20)"PERCENT OF TOTAL INCOME":PRINT
1306 GOSUB1200:PRINT
1308 GOSUB1400:PRINT
1310 FORJ=2TO18:READA$:IFD(J)=0THEN1330
1312 A=INT(D(J)/TI*300):PRINTA$;
1320 GOSUB1414
1330 NEXTJ
1340 FORJ=0TOQ:IFF(J)=0THEN1360
1345 A=INT(F(J)/TI*300):PRINTE$(J);
1350 GOSUB1414
1360 NEXTJ:PRINT
1362 GOSUB1400:PRINT:PRINT
1370 GOSUB1205:PRINT:PRINT:STOP
1380 OUT18,3:OUT18,17
1390 CONSOLE16:STOP
1400 PRINTTAB(20)"  -  -  -  -  5  -  -  -  - 10  -  -  -  - 15  -  -  -  ";
1402 PRINT"- 20  -  -  -  - 25":RETURN
1410 RETURN
1414 IFA<1THENPRINTTAB(20)"]":GOTO1430
1415 IFA<2THENPRINTTAB(20)"]]":GOTO1430
1416 PRINTTAB(20);:FORK=1TOA:PRINT"]";:NEXTK:PRINT
1430 RETURN
1600 OUT18,3:OUT18,17
1610 CONSOLE16
1620 STOP
1800 OUT18,3:OUT18,17
1810 CONSOLE18
OK
```

lieve the user of this calculation when inputting the account level, i.e., insurance is normally paid on an annual basis and yet the user must calculate and enter the amount for his budgeting period.

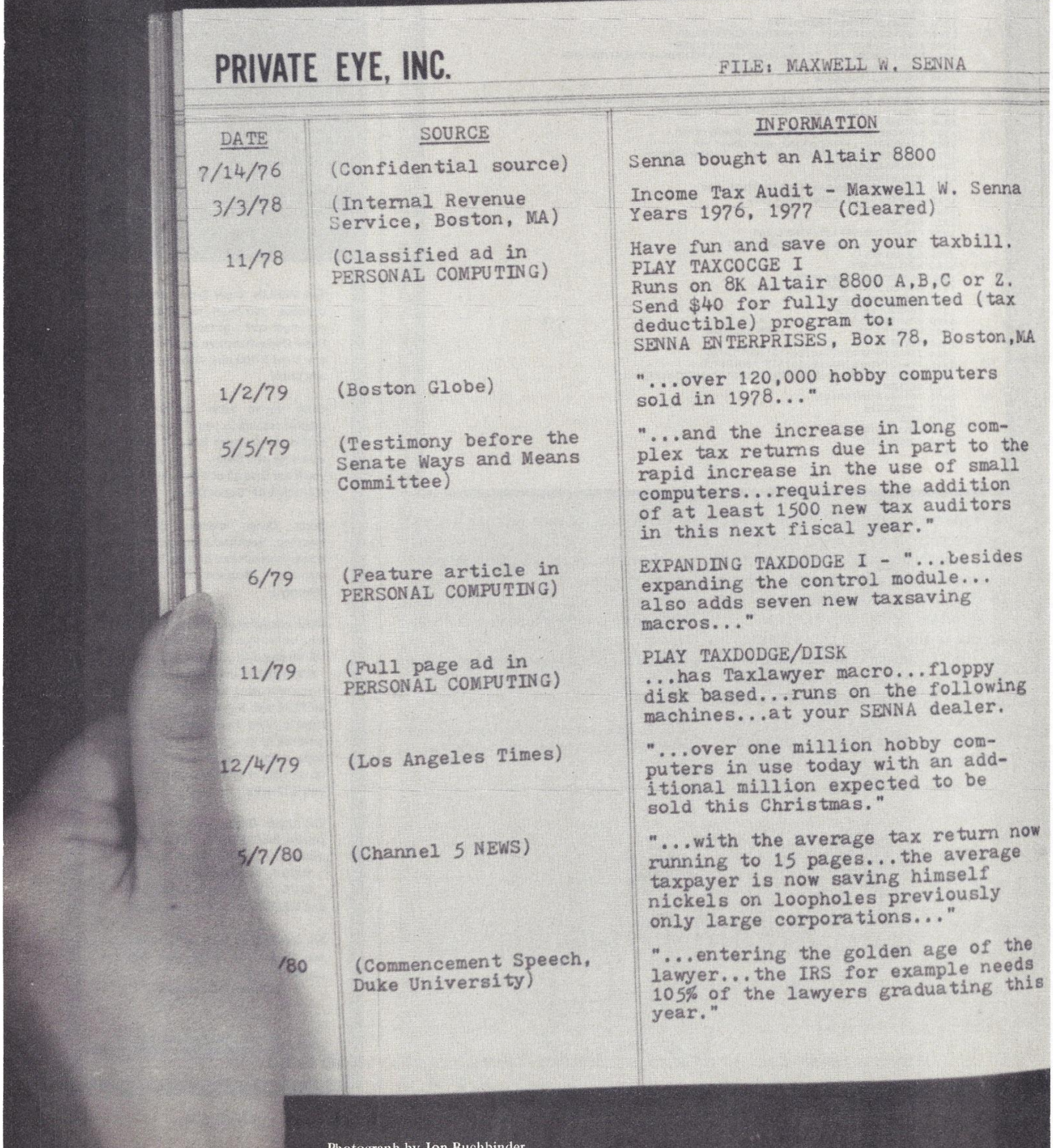
A further, rather substantial sequence to the program could permit the entry of actual expenses for the budget period and the automated adjustment of budget accounts for the remainder of the year. In addition, this sequence might usefully permit a thorough reworking of the annual budget as the figures for each expiring budget period become available, again employing the recursive technique illustrated in this program.

In any event, it must be evident that we are about through moving in 4K increments toward the limit of microcomputer core storage. Our sights must now be set for the floppy disk. And then???

O.E. Dial is a professor in the Univ. of Colorado Graduate School of Public Affairs, Boulder, Colo.

How do you want this sent — asynchronous or bisynchronous?

A LEMONADE

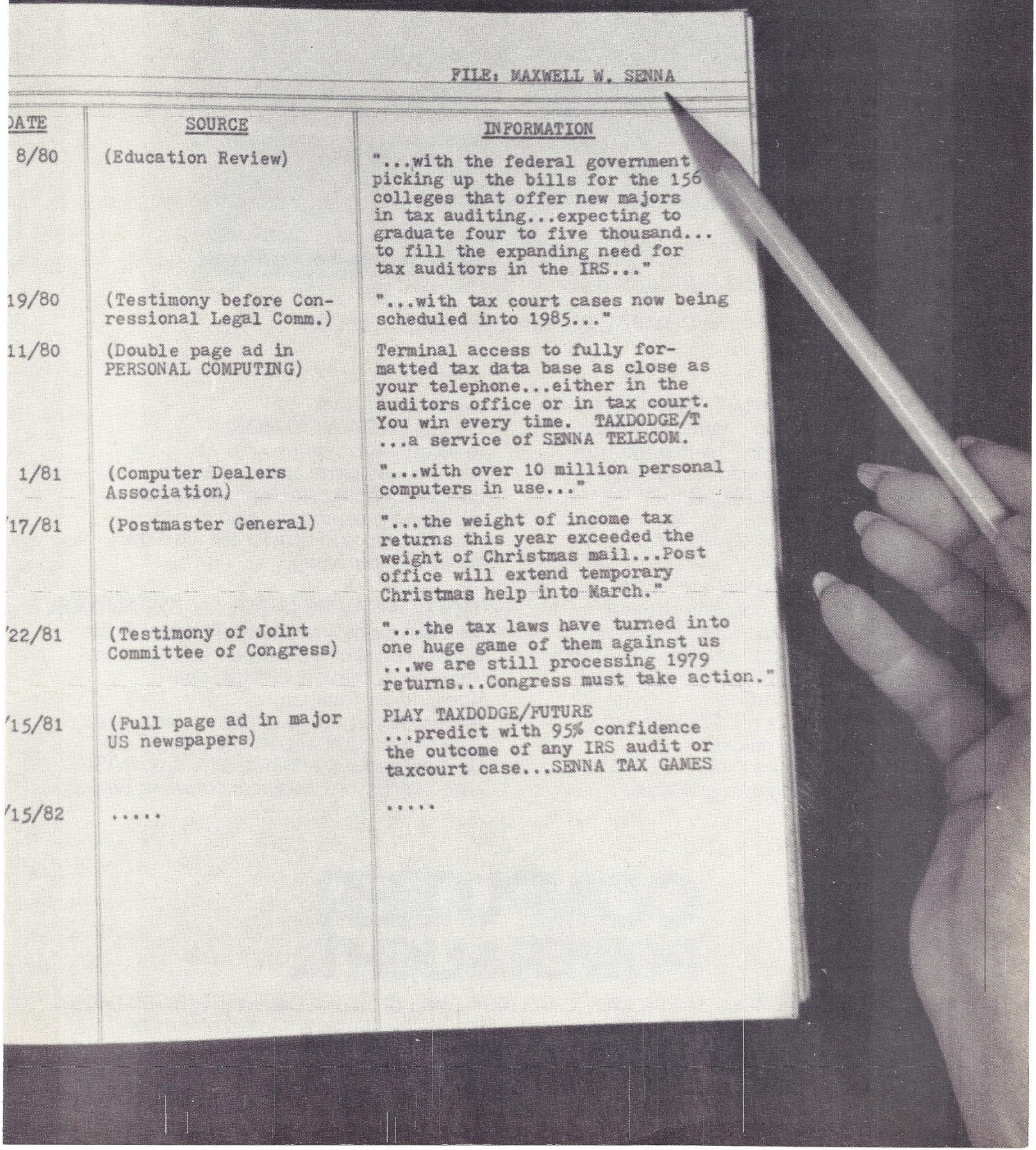


PRIVATE EYE, INC. FILE: MAXWELL W. SENNA

DATE	SOURCE	INFORMATION
7/14/76	(Confidential source)	Senna bought an Altair 8800
3/3/78	(Internal Revenue Service, Boston, MA)	Income Tax Audit - Maxwell W. Senna Years 1976, 1977 (Cleared)
11/78	(Classified ad in PERSONAL COMPUTING)	Have fun and save on your taxbill. PLAY TAXCOCGE I Runs on 8K Altair 8800 A,B,C or Z. Send $40 for fully documented (tax deductible) program to: SENNA ENTERPRISES, Box 78, Boston,MA
1/2/79	(Boston Globe)	"...over 120,000 hobby computers sold in 1978..."
5/5/79	(Testimony before the Senate Ways and Means Committee)	"...and the increase in long complex tax returns due in part to the rapid increase in the use of small computers...requires the addition of at least 1500 new tax auditors in this next fiscal year."
6/79	(Feature article in PERSONAL COMPUTING)	EXPANDING TAXDODGE I - "...besides expanding the control module... also adds seven new taxsaving macros..."
11/79	(Full page ad in PERSONAL COMPUTING)	PLAY TAXDODGE/DISK ...has Taxlawyer macro...floppy disk based...runs on the following machines...at your SENNA dealer.
12/4/79	(Los Angeles Times)	"...over one million hobby computers in use today with an additional million expected to be sold this Christmas."
5/7/80	(Channel 5 NEWS)	"...with the average tax return now running to 15 pages...the average taxpayer is now saving himself nickels on loopholes previously only large corporations..."
/80	(Commencement Speech, Duke University)	"...entering the golden age of the lawyer...the IRS for example needs 105% of the lawyers graduating this year."

Photograph by Jon Buchbinder

VISION



FILE: MAXWELL W. SENNA

DATE	SOURCE	INFORMATION
8/80	(Education Review)	"...with the federal government picking up the bills for the 156 colleges that offer new majors in tax auditing...expecting to graduate four to five thousand... to fill the expanding need for tax auditors in the IRS..."
19/80	(Testimony before Conressional Legal Comm.)	"...with tax court cases now being scheduled into 1985..."
11/80	(Double page ad in PERSONAL COMPUTING)	Terminal access to fully formatted tax data base as close as your telephone...either in the auditors office or in tax court. You win every time. TAXDODGE/T ...a service of SENNA TELECOM.
1/81	(Computer Dealers Association)	"...with over 10 million personal computers in use..."
17/81	(Postmaster General)	"...the weight of income tax returns this year exceeded the weight of Christmas mail...Post office will extend temporary Christmas help into March."
22/81	(Testimony of Joint Committee of Congress)	"...the tax laws have turned into one huge game of them against us ...we are still processing 1979 returns...Congress must take action."
15/81	(Full page ad in major US newspapers)	PLAY TAXDODGE/FUTURE ...predict with 95% confidence the outcome of any IRS audit or taxcourt case...SENNA TAX GAMES
15/82	.....	.....

SEE THE LIGHT

The COMPAL-80 computer system: **$2300.00**

WHAT YOU SEE IS WHAT YOU GET!
The computer, with 16K memory, input/output ports, keyboard, 9" video monitor, all enclosures, and powerful extended BASIC language – Just plug it in and you're ready to go!

WE'RE READY NOW.
The COMPAL-80 is a complete, fully-assembled guaranteed computer that is available now. Put it to work for you on home budgets, checkbook balancing, investment profiles, mailing lists, record/book/recipe indexes, text editing, mathematical/logical skill development, word games, strategy games, sports simulations, and more . . . We have ready-made programs available,

or if you'd rather,
WE'LL SHOW YOU HOW
to write your own programs!
Anyone can write and speak eloquently in computer language after attending our series of classes which deal directly and efficiently with the art of programming.

THE COMPAL-80 IS FULLY EXPANDABLE.
You can easily increase the power of the COMPAL-80 computer, by adding more memory, a printer, plotter, dual floppy disk drive, etc. . . .
Inquire about our business systems with dual floppy disk and printer, in a fully-coordinated system including disk BASIC and a variety of business software packages.

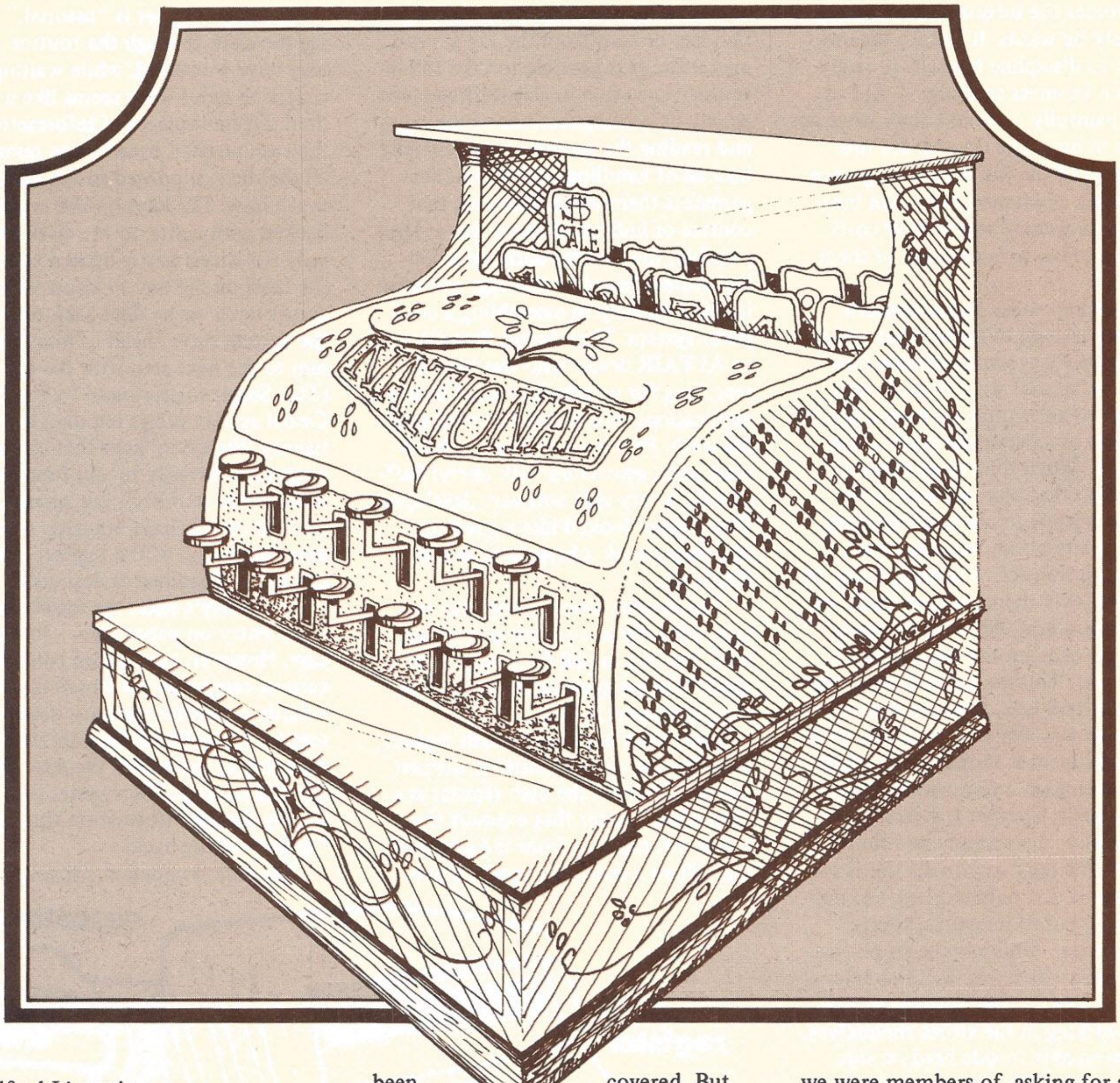
COMPUTER™
POWER & LIGHT INC.

12321 Ventura Blvd., Studio City, CA 91604 (West of Laurel Canyon) 213 760-0405
Open 12 to 10 Tuesday-Thursday, 12 to 5 Friday-Sunday (closed Mondays)

Bank financing arranged. Also Credit Cards and Cash.

CIRCLE 24

KEEPING TRACK



By Alfred Liquoris

Full computer control is coming to small retail stores. A Lemonade Opportunity lurks in this new area, though costs are still above the personal computing level. A lot of people can make money on this.

"Men's clothing is a highly seasonal business", says Larry Garcia, one of the principals in MR. CASUAL, a three-store chain in Albuquerque. "A few years ago, the store owner would just go down to the bank and borrow enough to stock up for fall. He'd order what he thought would sell, pay the bank back off the top of that sales income, and then make something for himself after that had been covered. But have you been paying attention to interest rates recently? You can't do that any more. If you borrow one dollar before you absolutely need it and if you keep it one day longer than you have to, you pay through the nose.

"The time has passed when the small store owner can operate on averages. He's got to have specific, detailed, current information on his market, stock, and operations if he expects to stay in business."

"We realized about two years ago that the only possible way to keep track of our business was to use a computer system, but we didn't know anything about them, so we began to write to the trade associations we were members of, asking for advice."

"Yes," says brother Manny Garcia," then we'd get a letter full of generalities and copies of some reports that didn't seem like anything we could use. They passed our name along to manufacturers, too. Pretty soon the field men for the big point-of-sale equipment companies would show up to tell us how to get computerized."

"The problem was that most of those guys were used to dealing with big department stores and didn't quite know what to do with a little operation like ours. We're doing only three-quarters to a million dollars a year altogether, and that isn't very interesting to a giant company. Usually the salesman would have a problem demonstrating his small business pack-

age to us and would begin to fumble through the operation manual. After a few of these meetings we realized that we'd have to do the job ourselves."

Almost any computer systems expert can do a competent job of giving a customer the information handling capacity he wants. It's up to the customer to discipline himself, to study his own business carefully – and usually painfully – so he knows what he wants to buy. The Garcias did this hard part of the job with a vengeance, rigorously examining their own business to determine where their costs were and how to keep track of them in detail.

Inventory was a major concern. "It isn't enough to know what came in two weeks ago and estimate how many we've sold since then. It isn't enough to know that we have a box of six shirts of a particular style, color and size. When somebody takes a shirt from that box, we want to be able to track it to the shelf, perhaps through alteration, to a customer, through a transfer to another store, to the shelf over there, or even back to the original box. We want to know when we ordered the shirt, who the buyer was, who the supplier was, who the salesclerk was, sometimes who the customer was. We want to know what our investment is in that shirt, whether it's eating up interest while it rests on the shelf, whether it is likely to cost money in alteration and finishing or not. The only way to do this is to keep track, not only of classes of merchandise, but of individual items."

The three Albuquerque stores handle roughly thirty-thousand individual items in a year. The business principals, working in the stores themselves, have been able to ride herd on this stock successfully, but expansion to just one more location changes that. Fortunately, there is now a good mechanism for tagging and working with large numbers of items.

The point-of-sale industry has developed a multiple-part tag that can be stamped with both printed and punched-hole codes representing all necessary information. MR. CASUAL has standardized on these tags and is equipped with punches to make them. The great virtue of the tag is that it's made of several identical sections. One section can be torn from the tag at each operation. It is possible to take full inventory in a store in about three hours, simply by pulling sections of the tags, and reading the sections automatically instead of handling and tallying the garments themselves. Knowing that control of individual items was at least possible, the Garcias outlined a control system based on the tags and went looking for help in assembling a computer system. They found Pete Conner, an ALTAIR dealer who was himself searching for worthwhile small business applications into which he could sink his teeth. Pete is an engineer, a hardware man, associated with Darryl Paffenroth, a very able software developer. This project looked like a good one and they struck a bargain with MR. CASUAL.

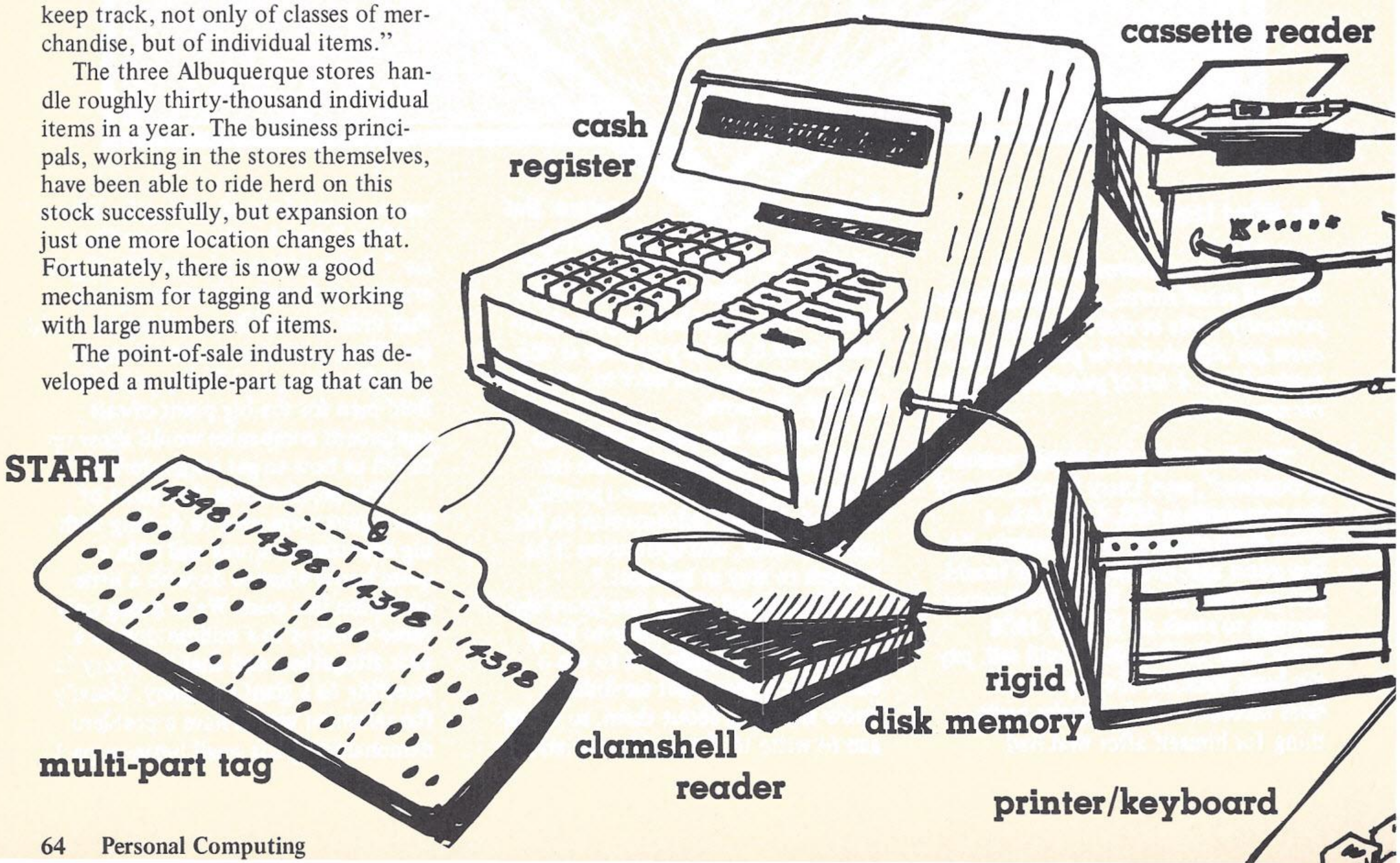
The system is now going into service, not only as a practical working tool, but as a model for a large class of small-business systems to come. The general scheme is this:

Each store has a single cash register, a big electronic beast stuffed with circuitry. Tied into the cash register is a "clamshell" reader that extracts the coded information from the punched tags. When a customer makes a purchase, the clerk pulls a section of the tag, feeds the data to the register, and punches in a set of additional data – his own identification, the date, store number, information on cash, credit, etcetera.

The cash register is "tutorial," leading the clerk through the routine. You may have wondered, while waiting for a clerk to enter what seems like a hundred different pieces of information into the cash register, how he can remember all that he is supposed to do. He cheats, that's how. The keys on the register are backed with lights, so the clerk must only see which key is lighted and note the label on the key to know what comes next. As he does each operation, the system stays ahead of him, leading him to the next step with the light. (This Singer register used by the Conner/Casual system offers ten digit keys and twenty "function" keys that can be assigned arbitrarily to any function – department number, cash, charge, tax, amount of purchase, transfer, whatever. Operation of the register itself can be programmed as desired.)

For safety's sake, the register prints every entry on paper tape – just in case. However, cumulative totals in various categories are stored in non-volatile computer memory inside the register (actually ferrite cores) for dumping at the end of the day. That's just the backup. The register constantly dumps information through a "spigot" at the back.

The data is coded in a special



Singer/ASCII format inside the register and is output as a non-computer-standard unit that converts that odd signal into another form (just the signal is converted, not the coded data), and records it on a digital cassette. All of the register entries are dumped onto this cassette through the day.

At day's end, the cassette is carried off to the computer (an ALTAIR 8800B) and read by a playback device that converts the *data* into RS232 (ASCII) format ...much handier for processing in the microcomputer. This converted data is dumped into a disc memory.

In this case, the memory is not a floppy disc, but a rigid disc with some ten million bytes of storage. A major task in development of the Conner/Casual system has been interfacing the rigid disc with the computer. The units were not designed to be compatible and interfacing is much more complex than simply matching plugs. Software and hardware both had to be developed and debugged. (For this reason, it is not convenient to substitute different components for those in the present system, which is tightly integrated.)

The computer, with a set of programs designed for the purpose, now can operate on the data, not only displaying inventory information, but handling general ledger, payroll, accounts receivable, and accounts payable. More, the system can report all sorts of useful statistics, on a daily basis if desired.

For example, it's easy to see if alterations at a particular store are running above or below the expected average cost. Ordinarily, management may notice such a variance on a sort of intuitive basis and inquire of the store manager why expenses are so high. "We've had a lot of suits back there this month," he may say. With the computer system, it's easy to check. If there weren't a lot of suits through the system, maybe the manager has something going with the seamstress...or he doesn't really know why costs are high and is just giving an evasive answer. Maybe somebody is misinforming him. In any case, the odd situation can be spotted quickly.

Similarly, the store's buyers have different taste and judgement based on imponderable factors. The computer statistics can show very rapidly just whose choice of merchandise is selling. They can show who does certain jobs well, so personnel can be shifted to their areas of strength. Also, the kind of merchandise that sells well at three stores in different areas of a single small city varies a great deal. Floral shirts may sell well at one store and not at all at another. Computer control allows rapid shifting of merchandise to take advantage of these differences ... with comparatively little "shrinkage" of stock owing to pilferage.

Statistics may point to faulty manufacture of garments that always seem to require more attention than others. If a particular line of suits, for example, always consumes ten to fifteen dollars in alterations, the store owner may get the manufacturer on the horn for a pointed discussion about either splitting the cost or taking a load of suits back.

The Conner/Casual system even allows management to model various possible courses of action based on the information presented by the process. Back to the matter of borrowing from the bank: the computer system can tell management how much it will cost to borrow for a purchase this week, when merchandise is available at a reduced price, compared to buying in three weeks, when the price will be back to normal. One can play "what if," experimenting in detail with complicated alternatives.

This Conner/Casual system, hardware and software together, is inexpensive only in comparison with more nearly conventional, but less effective small business computer packages. At about sixty-thousand dollars for the system with software, computer, video monitor, printer, keyboard, rigid disc and controller, cassette reader, three recorders, three registers, three clamshell tag readers, and a tag maker, the cost is half that of anything the Garcias have seen elsewhere ... and it will do their job. The system is not limited to the narrow business applications of particular concern. With resident extended BASIC the operator can do whatever he likes.

What has this to do with Lemonade?

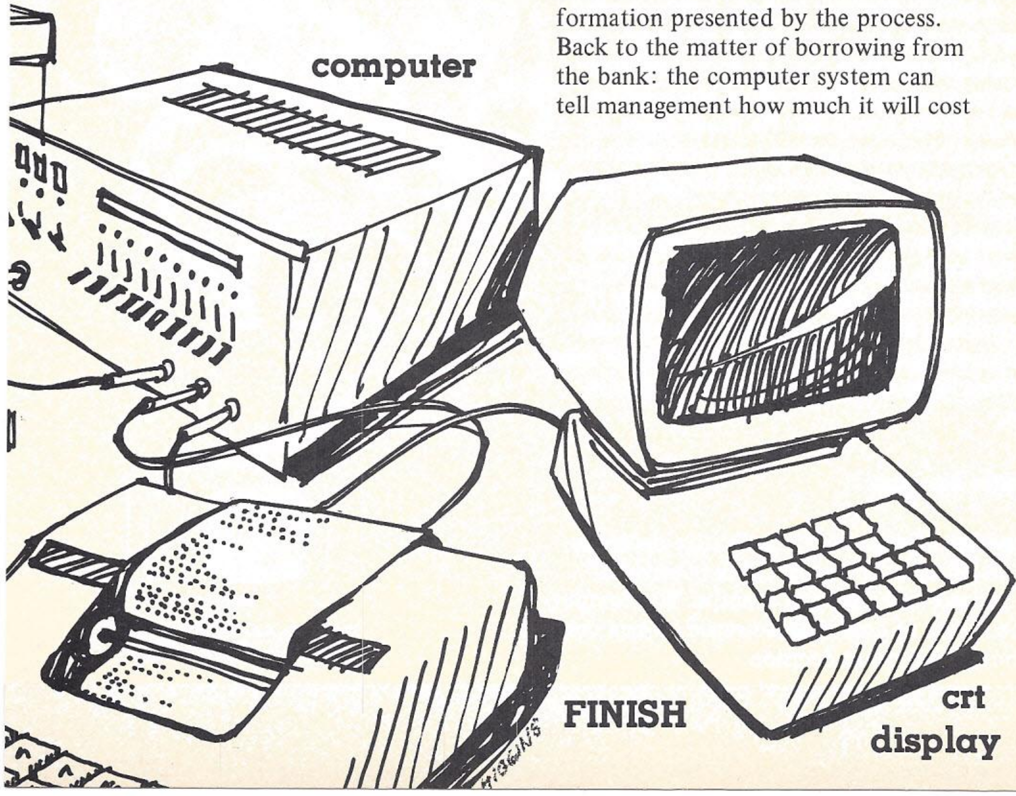
Well, notice that the registers are not on line with the computer. If the computer goes down (which it will, from time to time) the system as a whole doesn't drop dead in its tracks. Further, it may not be desirable for a small store to buy a complete system, but only to lease the register, reader, and recorder, sending the cassette tapes in to the computer operator at the end of the day.

A computer entrepreneur (or several, banded together in a small company) may provide the computer services to a number of small clients on a single system, sending out kids on bicycles to pick up the cassettes and deliver printed-out data.

As hardware costs diminish in the next months and years, small computer service companies can help small retail businesses of many types to gain tight, practical control over their operations. It won't be easy.

The agonizing, laborious self-examination the Garcias went through to learn what was really required for MR. CASUAL' S control must be repeated by others determined to gain control of their own situations.

The Conner/Casual system is a trailbreaker. Some number will surely be sold as complete systems, a number more adapted in various ways to different applications. More importantly, this successful effort should encourage development of a whole class of low-cost, effective business applications that spread significant computer power from big establishments to small ones.



YOUR PERSONAL GENIE

by Tom Munnecke

It helps you with your income tax, then it takes you in the Starship Enterprise on an outer space crusade against the Klingons. It teaches you Boolean logic, then it becomes an opponent at checkers. It draws vivid pictures on your television set, then telephones a distant computer to calculate the value of your personal stock portfolio.

What is this personal genie? How can it take on so many personalities? It is the personal computer, and its personalities are the unique products of its programmer. The computer is capable of nothing more, nothing less than the programmer instructing it. For all the precision and rigidity associated with a computer, the programmer's work is still a uniquely personal reflection of himself.

The fundamental connection between the programmer and the computer is the computer language. The increasing number and sophistication of computer languages bring the power of the personal computer to the non-professional.

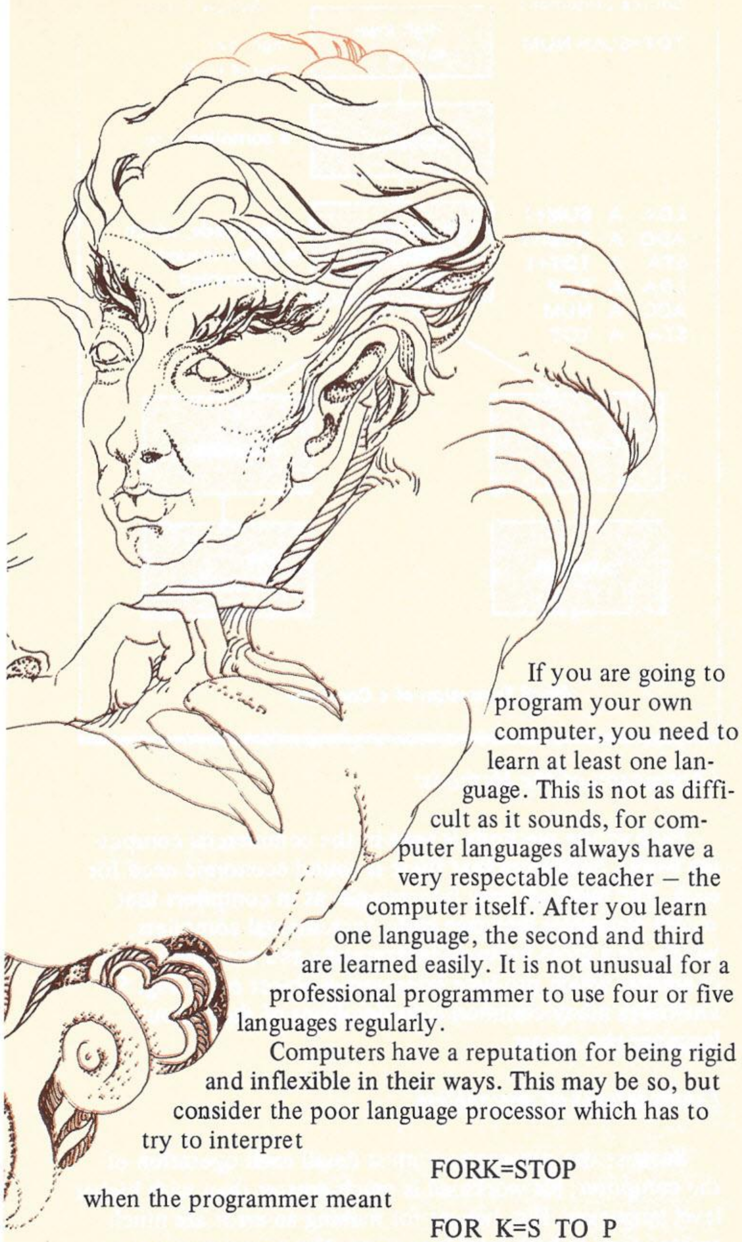
Computers are simple to deal with once certain fundamentals are understood. After that, learning becomes a trial and error experience. A person learning to walk does not need to understand each muscle, joint, and bone; he simply tries to walk and corrects his mistakes. So it is with computer programming. The novice programmer does not need to know the intricacies of the computer. He needs only: to know the fundamentals of the language, to know what his errors are and how to correct them, and to have time enough to try out his ideas.

The personal computer is a tool – the most powerful tool ever put in the hands of the private individual. Its potential is limited only by its owner's capacity to apply it.

This article provides a head start on learning any computer language, discussing the merits and drawbacks of many of the computer languages available to the personal computing enthusiast.

What is a Computer Language?

Computers operate in sequences of primitive decisions made in millionths of seconds. People think in terms of vague concepts derived over days and months. The computer language is the means of linking these vague human concepts to the primitive computer decision.



If you are going to program your own computer, you need to learn at least one language. This is not as difficult as it sounds, for computer languages always have a very respectable teacher – the computer itself. After you learn one language, the second and third are learned easily. It is not unusual for a professional programmer to use four or five languages regularly.

Computers have a reputation for being rigid and inflexible in their ways. This may be so, but consider the poor language processor which has to try to interpret

FORK=STOP

when the programmer meant

FOR K=S TO P

Most of the rigidity of the computer is there for a purpose. If you learn how they interpret things, some apparent inflexibility will fade away.

In order for the computer and the programmer to communicate, they must have some common physical medium for communicating. Usually, this is a keyboard/printer or video display. The programmer enters his programs in whatever language he is using, in his version of the language, known as the source language. He then asks a language processor to prepare it for the computer to process it.

There are two types of language processors – translators and interpreters. The translator accepts the source language and translates it to an object language, which is then loaded into the computer to be executed. Translators are further broken down into assemblers and compilers. The assembler is a means of manipulating machine-level operations for a specific computer, while the compiler translates higher-level, or more human-oriented languages. Interpreters execute the source language directly without the intermediate process of translating to an object language.

Languages are classified into two vaguely defined classifications: high-level and low-level. A low-level language is one in which each of the source code instructions corresponds to a machine-level operation. Source code in a high-level language may generate many machine-level instructions.

Assemblers, Compilers, and Interpreters

Each of the types of language processors has its merits and drawbacks – assemblers give the programmer great power but require very detailed instructions; compilers support higher-level languages, but sacrifice machine efficiency; and interpreters are easy to use, but are not as efficient as compilers.

Assemblers

The assembler is the simplest form of computer language. It accepts source code and translates it one-for-one into machine-level instructions or object code. Thus, the programmer has detailed control (and responsibility) of each instruction. For example, the programmer might write a line in assembler such as:

NEXT JSR INCHAR ; Jump to subroutine to get a character.

'NEXT' is a label for the line. 'JSR' is a mnemonic for the Motorola 6800 instruction 'Jump to subroutine'. 'INCHAR' represents the address of the subroutine to be used. ';Jump . . .' is a comment inserted by the programmer to explain the instruction for documentation.

The assembler (for the 6800) will assemble this instruction into the hexidecimal '8DXXXX' where '8D' is the operation code for branch to subroutine, and 'XXXX' is the address of subroutine INCHAR. See Fig. 1.

Since the assembler may not know where the INCHAR subroutine is to be located when the program is executed, it must be resolved at a later time by the *loader* program.

Compilers

The compiler acts much like the assembler, but works with higher level languages. The compiler understands more

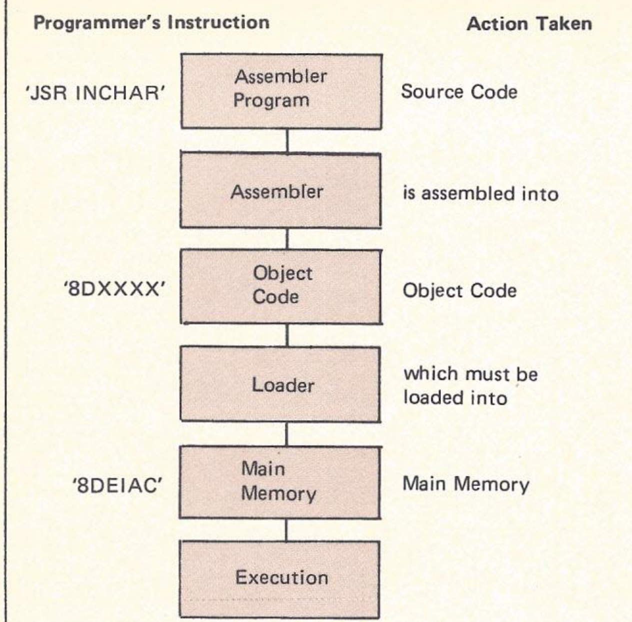


Fig. 1 Operation of an Assembler

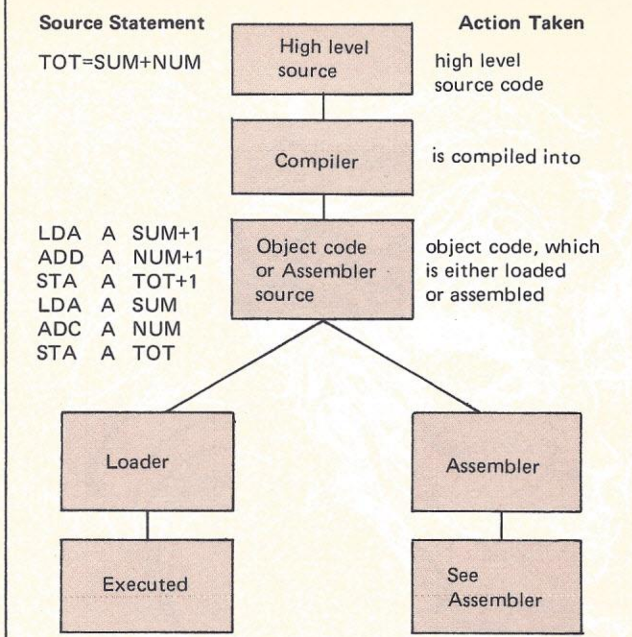


Fig. 3 Operation of a Compiler

complex expressions, and does much more work than the assembler. Figure 2 illustrates a single high level language expression which would require 6 lines to write in assembler.

Compiler	Assembler			
TOT=SUM+NUM	LDA	A	SUM+1	Add the right
	ADD	A	NUM+1	most bytes and
	STA	A	TOT+1	store the result.
	LDA	A	SUM	Add the left
	ADC	A	NUM	most bytes and
	STA	A	TOT	store the result.

Fig. 2 Comparison of high-level expression and its low-level language equivalent.

This is a simple example, but a more complex example, such as:

TOT=(SUM + NUM/1.238 * COS (ARC/360)) ** 2/7.32

could give the assembler language programmer a tremendous amount of difficulty.

Typically, a compiler produces assembly language code, which is then passed through the assembler.

Interpreters

The interpreter is a departure from the techniques of the assembler or compiler. While the translators create a program which must be loaded and executed later, the interpreter executes source instructions directly. The source remains in its original form.

Many languages may be either compiled or interpreted, although some features of a language may make compilation difficult, if not impossible. The interpreted language can change its interpretation as it receives new data, while the compiler does not know what data the program will receive until after it has finished its work.

Comparison of the Methods:

Each of the methods is used in the commercial computing world, indicating that there is sound economic need for each. The methods may intermingle, as in compilers that accept assembler language code, incremental compilers, which are a cross between interpreting and compiling, and compilers which produce interpretive object code. Fig. 5 illustrates many common considerations of the various language processors.

Disadvantages of Assemblers

Because the programmer must detail each operation of the computer, his workload is much greater than with higher level languages. His chances for making an error are much greater than in high-level languages. The programmer can easily become enmeshed in the maze of details he must remember. Modifying an intricate assembler language program may be very difficult, if not impossible. Assemblers are not usually interactive, requiring the entire program to be reassembled when an error is made.

Advantages of Compilers

The compiler is capable of supporting much higher level languages than assembler or macro assembler. The programmer can work faster, make fewer errors, and learn the language faster than he can assembler. The compiler's object code may be executed much faster than an interpreter could execute the program (between 5 and 10 times faster). Programs written in the higher level language may be recompiled on a new type of computer, without modifying the program.

Disadvantages of Compilers

Compilers are usually large, complex programs which require some time to compile a program, in addition to a significant amount of off-line storage. Compilers are not

usually interactive, because they require an entire program to be recompiled when a single change must be made.

Due to the internal workings of the compiler, data types must be fixed during compilation. This process, known as binding, reduces the program's ability to adapt to new data as the program is executed. An interpreter, however, does not bind its variables until execution.

Advantages of the Interpreter

Since the interpreter executes its source code directly, the programmer may interact more directly with the computer. Usually, the interpreter provides a direct mode, where the programmer may execute statements directly as he enters them, and an indirect mode, where his commands are stored in a program for later execution. The programmer can usually stop the program, examine variables, and resume execution. Some interpreters (such as APL and MUMPS) provide an EXECUTE command, which allows the program to execute a character string as if it were program text. Conversely, some interpreters (MUMPS) allow a program to treat its own text as data. Interpreters are useful for systems where the language processor needs to be 'built in' to the computer, as in intelligent terminals.

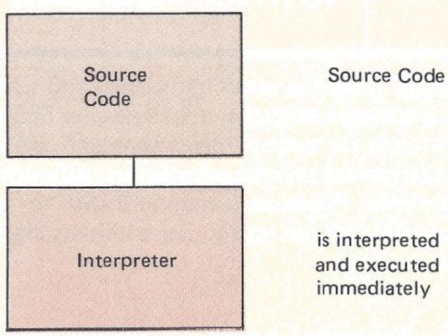


Fig. 4 Operation of an Interpreter

Characteristic	Compiler	Interpreter	Assembler
Binding Time	Compile	Execution	Assembly
Off line Storage	much	little/none	much
CPU Efficiency	medium	low	high
Programmer Efficiency	medium	high	low
Program Size	large	medium	small
Error Detection Language	machine	source	machine
Interactive Debugging	no	yes	no
Language Processing Efficiency	low	medium	high

Fig 5 Comparison of the features of the various types of language processors.

• Binding time – the point when the program's data types are fixed. • Offline Storage – the amount of storage such as floppy disks, cassettes, etc, required for the language processor to work. • CPU Efficiency – of the program being processed. • Programmer Efficiency – of the programmer writing the program to be processed. • Program Size – of the object code, or source code, in the case of the interpreter. • Error Detection Language – the language in which run time errors are detected.

Disadvantages of the Interpreter

Interpreters tend to be slower than compilers, between 5 and 10 times slower, as a rule of thumb. This slowness is due to the interpreter's need to analyze each statement every time it sees it, whereas the compiler need analyze it only once. The interpreter program must remain in memory for even a small program.

A Bit of History

The first computers were large, expensive devices requiring a roomful of air conditioning just to keep them cool. Programming them was very difficult, and they ran quite slow:

> ". . . the machine will then continue in operation hour after hour, completely checking its own results until either the problem is solved, or a breakdown occurs" (A Manual of Operation for the Automated Sequence Controlled Calculator, Harvard University, 1946).

At that time, a computer cost millions of dollars, and a programmer cost a few hundred dollars per month. Today, a computer costs hundreds of dollars, and the programmer costs thousands of dollars per month. To put it in another way, in 1946 a computer cost the equivalent of 250 programmers, today the programmer costs the equivalent of 100 computers.

Everyone agrees that computers should be used 'efficiently'. The problem is that people think of making the CPU efficient, not the person using it. The microcomputer has undermined the conventional wisdom of computer efficiency. The person who spends several month's rent on a personal computer wants to see it do something for him immediately, regardless of whether it uses the CPU 'efficiently'. Chances are he uses the computer only a few hours a day. On the other hand, the professional programmer who works as one of a score of programmers using a large computer must contend with CPU efficiency in order to keep from overloading the computer.

The microcomputer user needs to worry about CPU efficiency only when he reaches some limit – not enough memory response not fast enough, etc. Since no one else is waiting to use his computer, he does not have to worry about inefficiencies which do not force him beyond his limits.

The large computer programmer, however, must constantly worry about sharing the computer with all the other users. Even if a program works fast enough for him, and uses little enough memory, it still must be made 'efficient' for the other users of the system.

As a result of this historical concern for CPU efficiency, people are fixated on "making the computer run efficiently". Language design has been heavily weighted in favor of making the computer efficient, not the programmer.

The personal computing software scene was a completely unforeseen turn of events. None of the language designers ever thought that the programmer would be working alone on his own computer. As a result, the design tradeoffs were heavily slanted in favor of the commerical user.

Which language is Best?

> "I speak Spanish to God, Italian to women, French to men, and German to my horse".
> Charles V of France

What is the best language? BASIC? Assembler? PL/M,

PROGRAM
- LINE
  - LINE NUMBER
  - COMMANDS
  - ARGUMENTS

Fig. 6 BASIC Program Elements

PROGRAM
- STATEMENTS
  - LABEL
  - OPERATION CODE
  - OPERANDS

Fig. 7 Assembler Program Elements

PROGRAM
- BLOCK
  - STATEMENTS
    - EXPRESSIONS

Fig. 8 PL/M Program Elements

PROGRAM
- ROUTINE
  - LINE
  - LABEL
  - COMMAND
  - ARGUMENTS

Fig. 9 MUMPS Program Elements

WORKSPACE
- FUNCTION
  - LINE
    - OPERATORS
    - LITERALS
    - FUNCTION REFERENCES

Fig. 10 APL Program Elements

PROGRAM
- GROUP
  - LINE NUMBER
  - COMMAND
  - ARGUMENTS

Fig. 11 FOCAL Program Elements

MUMPS, APL, PASCAL, FORTRAN, SNOBOL, COBOL, LISP, COMIT, MAD, or any of the hundreds of others? And after the best language is chosen, which dialect is best? Consider the dialects of BASIC: Tiny BASIC, Extended BASIC, BASIC Plus, Business BASIC, ANS BASIC . . .

Perhaps a good analogy could be drawn between computer languages and spoken languages. Which spoken language is best? English? French? Chinese? Italian? It all depends on what you want to do with it. If you are in Paris, French would be a good contender for the 'best' language. Suppose you are in Kansas, and believed Charles' statement above that Italian is best for speaking to women. Romantic pretentions aside, you would probably have better luck with English.

The "best" computer language is not selected on the basis of its syntax or grammar. It is a very pragmatic decision based on what is available, what the programmer knows, whether it can perform the task at hand, and what programs are available to him from other sources.

The selection of a computer language is an important decision to the personal programmer for many reasons beyond the above pragmatic ones. The language a programmer uses profoundly affects the way he sees a problem. As Whorf said. "We dissect nature along lines laid down by our native language". The APL programmer thinks in terms of vectors, the MUMPS programmer thinks in terms of data bases, and the Assembly language programmer thinks in terms of individual bytes of memory.

Therefore, in reviewing each of the languages, the reader must apply them to his own needs. The following list is a sample of some of the languages available (or may be soon) to the micro-computer user.

BASIC – (Beginner's All purpose Symbolic Instruction Code). This is the most common high-level language used on personal computers. It is a very simple, easy to learn language. There is a large library of programs available, since BASIC is used in many universities and schools. Because it is a simple language, it is somewhat limited and difficult to use for some complex problems. BASIC is usually interpreted on microcomputers, although some compilers exist. Programs written in BASIC for one computer can often be run on another with only slight changes.

Assembler – Assembler language is commonly used on personal computers. Since many personal computers have neither the memory or Input/Output capability to run an assembler, the programmer often manually assembles his program and enters it through the switches on the panel. Assembler language is unique to each computer, so program exchange is limited to one particular computer type.

Assembler language is the common denominator of all programs – eventually, all programs are just a sequence of assembler-level instructions. Therefore, any one wishing to really know how his computer works must learn at least a little Assembler. Often, a program is written in a high-level language which calls an Assembler language subroutine for difficult or critical portions of logic. This can be a very economical mix for programs which exceed the limits of a high-level language.

PL/M – (A program name copyrighted by Intel Corp.) is a compiled language derived from IBM's PL/1. Versions exist for the 8080, 6800, and Signetics 2650. Some high speed, mass storage (floppy disk, for example) is required. It is an alternative to assembler, producing slightly less efficient programs in much less programming time. A basic user would find PL/M difficult to use for simple problems, but easier to use for more complex problems. There is no extensive library of programs in PL/M as with BASIC.

XIMEDIA PRESENTS

The SOROC IQ120

CURSOR CONTROL. Forespace, backspace, up, down, new line, return, home, tab, PLUS ABSOLUTE CURSOR ADDRESSING.

TRANSMISSION MODES. Conversation (half and full Duplex) PLUS BLOCK MODE — transmit a page at a time.

FIELD PROTECTION. Any part of the display can be "protected" to prevent overtyping. Protected fields are displayed at reduced intensity.

EDITING. Clear screen, typeover, absolute cursor addressing, erase to end of page, erase to end of line, erase to end of field.

DISPLAY FORMAT. 24 lines by 80 characters (1,920 characters).

CHARACTER SET. 96 characters total. Upper and lower case ASCII.

KEYBOARD. 73 keys including numeric key pad.

REPEAT KEY. 15 cps repeat action.

DATA RATES. Thumbwheel selectable from 75 to 19,200 baud.

SCREEN. 12 inch rectangular CRT — P4 phosphor.

SPECIAL INTRODUCTORY PRICING

Kit $ 995.00 **Assembled $ 1,295.00**

(Price includes block mode, lower case and 24 line options.)

Specials of the Month

North Star **Micro Disk** with power supply and cabinet	Kit — $699
	Assembled — $799
IMSAI I-8080 with **TDL** ZPU	Kit — $825
	Assembled — $975
Digital Systems FDS Disk Drive with **CP/M** Software (assembled only)	Single — $1,750
	Dual — $2,350
Mountain Hardware **PROROM**	Kit — $145
	Assembled — $210
Vector Graphic 8K RAM	Kit — $235
	Assembled — $285

NOW WE'RE TOLL FREE

800-227-4440

(in California, Hawaii, and Alaska, call collect: 415/566-7472)

XIMEDIA OFFERS A FULL RANGE OF PRODUCTS FOR THE PERSONAL COMPUTER ENTHUSIAST AND THE SMALL SYSTEM DESIGNER. LET US QUOTE ON ALL YOUR HARDWARE AND SOFTWARE NEEDS.

OUR RETAIL STORE — THE COMPUTERIST™ — IS NOW OPEN IN SAN FRANCISCO. CALL US FOR DIRECTIONS.

XIMEDIA

1290 24th Avenue, San Francisco, CA 94122
(415) 566-7472

COD orders freight collect. Orders with payment shipped prepaid. California residents add 6% sales tax. Please allow 3 weeks for delivery.

CIRCLE 25

MUMPS – (Massachusetts General Hospital Utility Multi-Programming System) is an interpretive language oriented towards interactive data management applications. MUMPS has many characteristics of BASIC, FOCAL, and IBM's PL/1. It differs from all these in that it has built-in data base capabilities for handling data on mass storage devices. Although not widely available on microcomputers now, the National Bureau of Standards published a standard version (NBS Handbook 118) which details how one would write an interpreter for MUMPS.

MUMPS has extensive data handling capabilities, suited for applications such as personal accounting, word processing, and general information systems. Since the development of MUMPS was federally supported, much MUMPS software is in the public domain.

APL – (A Programming Language) is a computer language derived from Iverson's elegant mathematical notation. It is a very powerful mathematical tool, having primitive functions for matrix inversion, inner products, sorting, and many other areas. Although initially developed for large scale computers, it is now available for portable commercial computers. APL is usually interpreted, and therefore well suited for interactive personal computing.

FOCAL – (Formulating On-Line Calculations in Algebraic Language) is a language brought out as an early on-line language for calculations. Its syntax is similar to MUMPS, although its functions are closer to BASIC. FOCAL is available on the 8080 and has a modest programming library.

Learning a Computer Language

Your first task in learning a new language is to build up a basic understanding of the language. This can be gained from the reference manual for the language distributed with the software. Magazines such as Personal Computing carry many articles on the more popular languages. There is a variety of books available in libraries and computer stores, and more advertised in professional data processing magazines.

When studying a language, it is helpful to divide the project into three areas:

SYNTAX – How you say something
SEMANTICS – What you mean
PRAGMATICS – How you make the language do what you want

Syntax. The syntax of the language is usually the quickest part to learn. How does the language distinguish between a number and a variable? Do you need a number before each line? What characters are allowed by the language?

Semantics. The semantic aspects of the language are more difficult to learn, but you do not have to understand everything to use the language. What are arithmetic functions in the language? How do you retrieve data from the terminal? How do you format output?

Pragmatics. This is the most difficult portion to learn, yet it is the skill most easily carried over to other languages. How do you make the language solve your problem? How do you create, change, and delete programs? Can you stop the program while it is executing, examine the state of things, then resume execution?

These three classifications are very useful for comparing languages. For example, BASIC, FOCAL and FORTRAN have similar semantics but different syntaxes. MUMPS and FOCAL have similar syntaxes, but different semantics.

With this background, you should be able to modify a simple program to make it do increasingly complex tasks. Each time you modify the program, use some new aspect of the language, being careful to add one aspect at a time. Then, try the new version to see if it does what you expect.

Each step of the way, you will be informed of your mistakes by your friendly adversary, the computer.

The Importance of Making Errors

> "Nine times out of ten, in the arts as well as life, there is actually no truth to be discovered; there is only error to be exposed."
>
> H.L. Mencken

Making an error in a computer program is a fundamental source of learning. You tried something and the computer told you it didn't work. The programmer who proudly announces "my last program worked the first time without any bugs" is a programmer who probably did not learn anything new writing it.

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Integer (16 Bit)	X	X	X	X	X	X
Byte			X		X	
Character String	X	X				
Floating Point	X	X	X	X		X
Logical		X	X			X
Labels		X				

Fig. 12 Cross Index of Data Element Types

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Assignment	LET	SET	←	SET	=	=
Read from Console	INPUT	READ	←□	ASK	INPUT	INPUT
Write to Console	PRINT	WRITE	□←	TYPE	OUTPUT	OUTPUT

Fig. 13 Cross Index of Data Movement

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
Unconditional Branch	GOTO	GOTO	→	GO	GOTO	GOTO
Conditional Branching	IF	IF	→	IF	IF	IF
Involation	GO SUB	DO	NAME	DO	CALL	CALL
Return from Involation	RETURN	QUIT	→0	QUIT	END	RETURN
Looping	FOR/NEXT	FOR		FOR	DO	DO

Fig. 14 Cross Index of Control of Flow

REVOLUTIONARY NEW METHOD OF... STORING, USING AND ORDERING DISKETTES ★

THE K-10 HOLDS 10 DISKETTES FIRMLY DURING SHIPMENT
YOU GET ONE K-10 FREE WITH EVERY 10 DISKETTES AT $47.50 TOTAL

RUGGED, DURABLE DESIGN... The Kas-ette/10 is made of strong and durable polypropylene to withstand rugged handling, storage, and shipping ... yet, its soft leather-type texture and book shape offers an aesthetically appealing case suitable for desk top, bookshelf or library. The library case provides a safeguard against environmental contaminants such as dust and debris which may affect the reliability of recorded data. It offers excellent temperature and humidity protection for intermediate or archival storage and shipping.

TRANSPARENT "FLEXIBLE FAN TABS"... The clear polystyrene insert with "flexible fan tabs" allows an operator to conveniently browse through all stored Diskettes, easily read labels, and to readily remove and replace Diskettes from the holder.

AN ORDERLY LIBRARY SYSTEM ... By using a coded label on the back spline and a coded label on each diskette, a library system can be maintained. When the library becomes large enough, a "Master File System" can be established to maintain an orderly library. The Master System consists of 100 coded ledger cards with titles for 1000 diskettes stored in one Kas-ette/10.

CONVENIENT "POP-UP" FEATURE ... The "pop-up" feature of the insert places the Diskettes conveniently in reach of an operator while sitting at a desk, terminal, or minicomputer system.

CALL NOW FOR THIS GROUP/3 SPECIAL
TOLL FREE (800) 423-5148

In California Call (213) 884-6678 Collect

★Fully compatible with systems and drives manufactured by IBM and such leading independents as DEC, Datapoint, Shugart, Control Data, Pertec, Memorex and others. Specify drive manufacturer and model number.

Price includes shipping in the continental U.S.

- Shipped prepaid in the continental United States
- Shipped air freight collect to Hawaii, Alaska, Puerto Rico and the Virgin Islands

GROUP/3®

P. O. Box 1452, 21050 Vanowen Street
Canoga Park, CA 91304

GROUP/3 is a part of Informatics Inc., the world's leading supplier of software products. We are a subsidiary of The Equitable Life Assurance Society of the United States, third largest life insurance company in the country.

®GROUP/3 and Informatics are Registered U.S. Trademarks

ALL PRICES SUBJECT TO CHANGE WITHOUT NOTICE

CHOICE OF COLORS: ☐ **BEIGE** ☐ **CHOCOLATE**
Diskettes are for System/3 ☐ System/32 ☐ Other ☐
K-10 with 10 Diskettes $47.50 each set
Manufacturer ______ Model No. ______
Quantity ______ Price ______ Total ______
Add $3.00 per order for handling unless you are enclosing payment.
(California residents add 6% sales tax)
Name ______
Firm ______
Purchase Order No. ______
Address ______
City ______ State ______ Zip ______
Tel. ( ) ______
Prices subject to change without notice.

	BASIC	MUMPS	APL	FOCAL	PL/M	FORTRAN
And		&	∧			
Or		!	∨			
Not		'	~			
Greater Than	>	>	>	*		
Less Than	<	<	<	*		
Equal	=	=	=	*		
Not Equal	<>			*		
Less Than Or Eq.	<=		≤	*		
Greater Than or Equal	=>		≥	*		

Fig. 15 Logical and Arithmetic Comparison Function
*handled by IF statement structure.

	BASIC	MUMPS	APL	FOCAL	PL/M
Addition	+	+	+	+	+
Subtraction	—	—	—	—	—
Divide	/	/	÷	/	/
Multiply	*	*	X	*	*
Exponentiation	↑		*	FExP	
Square root	SQR		*.5	FSQT	
Cosine	COS		2○	FCOS	
Tangent	TAN		3○		
SINE	SIN		1○	FSIN	
e^X Exponential	EXP		*	EXP	
Natural log	LOG		⍟	FLOG	
Absolute Val	ABS		|	FABS	
Greatest Integer	INT		⌊	FITR	
Random Number	RND	$R	?	FRAN	
Signum	SGN		X	FSGN	
Modulo		#	|		

Fig. 16 Cross Index of Arithmetic Functions

The absence of an error when writing a program indicates only that a situation new to the programmer did not come up – not that the programmer has learned the language.

There are generally four types of errors: syntax, semantic, pragmatic, and covert.

Syntax errors

The syntax error is the most common error which faces the beginning programmer. A syntax error is a statement that violates the language's basic rule for expression. Typically, they are caused by:

a) typing errors – a finger slips to the wrong key, a zero instead of the letter O, etc.

b) misunderstanding the syntax. The new programmer may not understand that he has to put a comma between variables in a print statement, or put apostrophes around literals.

c) confusing the syntax. The programmer might confuse a colon and the comma, or, he might carry over some syntax from another language he knows.

One thing in common with all these errors is that the computer can detect them. In most interpreters, the programmer may directly enter and execute any questionable statements. The lesson is clear: When in doubt, try it. Let the computer tell you whether it will accept the statement. Many manuals are not reliable enough to trust anyway.

The above advice flies directly in the face of conventional computer programming wisdom. In the past, there was considerable stigma attached to anyone found 'letting the computer do his debugging'. The theory was, that the computer is a valuable resource, and that a programmer should not waste computer time. Instead, he should carefully desk-check his program before each submission. In the microcomputer world, this philosophy is radically altered. It makes no sense for the programmer to check his work on paper when his computer is waiting for him to enter it.

Semantic errors

These errors are also common in the early stages of learning a new language, but continue to plague the programmer throughout the use of the language. These errors are statements which are syntactically correct, but do not perform the function desired by the programmer. Some typical semantic errors are:

a) Mode errors – the programmer tries to add a number to a character string, but the language does not handle the conversion.

b) Binding errors – the programmer names the wrong variable, label or subroutine.

c) Juxtaposition or sequencing errors. An end of a loop is placed too far down in the program, or a variable is used before it is initialized.

Most of the same advice for syntax errors applies to grammatical errors. Sometimes, grammatical errors can slip through and only be detected by erratic program behavior.

Pragmatic errors

The pragmatic error is a statement which is syntactically and semantically correct, but does not do what the programmer wants it to. These cannot be caught by the language processor. Typical pragmatic errors are:

a) wrong function or command – the programmer uses a sine function instead of cosine.

b) an improper formula – the programmer thought that Interest was Principal divided by Rate instead of Principal times Rate.

Pragmatic errors tend to be the last errors in a program to be detected, if only because the programmer will not see them until he cleans up the syntax and semantic errors and the program executes.

Pragmatic errors can be very difficult to detect, particularly in programs which are time dependent or involve much

	BASIC	MUMPS	APL	FOCAL	PL/M
Search		$FIND	?		
Extract	MID$ LEFT$ RIGHT$	$EXTRACT	SUBSCRIPTS		
Concatenation		_			
Convert String to Number		$ASCII			
Convert Number to String		$CHAR			
Length	LEN$	$LENGTH			

Fig. 17 Cross Index String Functions

logic. Pragmatic errors are generally discovered with what the computing world euphemistically calls "testing". "I'll test this program to make sure it won't blow up," is an often heard phrase. Unfortunately after he completes his testing, he all too often says "my program blew up".

Testing can confirm the existence of an error, not that one doesn't exist. Just because 99 combinations of input data were tried does not guarantee that the hundreth combination will not fail.

Covert errors

When a program is tested and declared correct by the programmer, any remaining errors are by definition covert. These are insidious problems that appear only when events combine to form some previously untried condition. Some covert errors are:

a) An angle in a trigonometric equation goes to zero, causing a zero divide error in a later division.

b) Improper data is entered, which the program does not reject as invalid. Recently, a program sent out a letter to the Emmet County Jail, "Dear Emmet C. Jail, you are among a select group of persons . . . " As the saying goes – garbage in, garbage out.

c) The programmer leaves room for only 3 digits of a number, but the number grows past 999.

Covert errors always have and always will exist in computer software. However, a great deal of attention in computer science circles has been given to writing programs which may be "proved" correct. These efforts, named "structured programming", "software engineering", and "composite design" will be covered in a future article. The fundamental principles common to these are:

a) Break the big problem into clusters of independent little problems.

b) Link the clusters together in a hierarchical manner such that each cluster is independently testable.

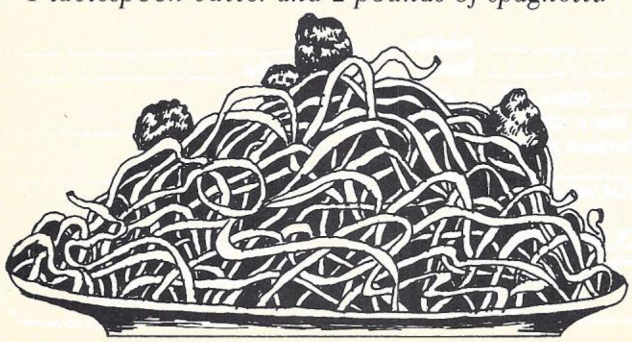
c) Limit the number of paths the program may take. This is accomplished by limiting the use of the GOTO statement.

The programmer should learn to improve his skills by analyzing the errors he makes. When he meets that benevolent dictator of linguistic purity – the error message – he should treat it as a means of learning a little more about the language.

SPAGHETTI BASIC

Suppose you are a chef at a large hotel and your most frequent banquet dish is spaghetti. You have a favorite recipe for Italian spaghetti that serves six people. The problem is that you normally serve anywhere from 7 to 550 people. Just calculating the correct amount of ingredients for your recipe can take you away from the kitchen for 30 minutes.

The ingredients in your spaghetti recipe serving six people are ¼ cup olive oil, 1 large onion, ¼ cup chopped parsley, 1 teaspoon oregano, 1 pound of ground beef, 1 pound 4 ounces of tomatoes, 6 ounces tomato paste, ¾ cup water, 1 bay leaf, 1 clove garlic, 1-½ teaspoons salt, ¼ teaspoon pepper, 1 tablespoon butter and 2 pounds of spaghetti.



Defining the Problem

In Lesson One (January/February PERSONAL COMPUTING) we said the first step in writing a program is defining the problem.

Since we want to know the amount of ingredients needed for any number of people we need a formula for converting each ingredient from six people to X people. Let's take a look at the olive oil. The recipe calls for 1/4 cup olive oil for six people. How much olive oil would it take for 25 people.

The computer could solve this by first determining the amount needed for one person and then multiplying that amount by 25. In BASIC this computation is expressed like this: .25/6*25 where .25 represents 1/4, (/) means "divided by" and (*) means multiply.

Of course, there are other formulas for solving this problem. Some of them may be more efficient than the ones above but since the computer will solve this equation in a matter of a few microseconds (millionths of a second), it really doesn't matter.

By simply substituting a variable (Lesson One) for the value 25, we have a formula for determining the amount of olive oil needed for any number of people. The formula is: .25 divided by 6 times P (number of people).

Since the problem with all the ingredients is the same as the problem with olive oil, our formula will hold up throughout the program.

Inputting the Data

The second step to writing a program is inputting the data. The only data we have is the proportions of ingredients, and the number of people we wish to serve. Calling the amount of olive oil "A", the amount of onions "B", the amount of parsley "C" and so on, we can enter the ingredient data into the computer with a READ and a DATA statement (Lesson Two):

```
10 READ A,B,C,D,E,F,G,H,I,J,K,L,M
20 DATA .25,1,.25,1,1,1.25,6,.75,1,1,1.5,.25,2
```

Note that we converted 1 pound 4 ounces of tomatoes to 1.25 pounds (earlier we converted 1/4 cup olive oil to .25 cup). You can easily use your computer to convert fractions to decimals by using the PRINT command followed by the fraction. To convert 1/4 to a decimal, simply enter:

```
PRINT ¼
```

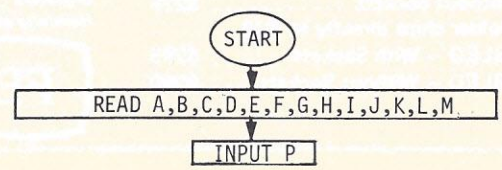
followed by a RETURN and .25 will be instantly printed on your video screen or teletypewriter. You can use this method to convert any fraction to a decimal.

By using a READ and DATA statement, we can enter all the ingredient data. Now, we need a method of entering the number of people we want to serve.

In our formula, the number of people is represented by the variable "P". By using an INPUT statement (Lesson Two), you can make this variable any number you want:

```
30 INPUT P
```

A flow diagram of our program at this point looks like this:



Notice that we don't include the DATA statement in this diagram. That is because a READ statement implies a DATA statement.

Computing the Data

Now that we have devised a means of INPUTTING THE DATA, we have to instruct the computer what to do with it. When we were defining this spaghetti problem, we said that the procedure for determining the amount of an ingredient is: amount of ingredient divided by 6 times the number of people (P).

In the case of olive oil, we assigned the variable A in the READ statement to represent the amount of olive oil. In BASIC language, a statement for the computation of olive oil is:

```
40 PRINT A/6*P
```

Likewise, a statement for figuring the amount of onions is:

```
50 PRINT B/6*P
```

and so on. We have now established a format for computing the amount of each ingredient. This is the only computation needed to solve our problem.

Our flow diagram now looks like this:

```
START
  |
READ A,B,C,D,E,F,G,H,I,J,K,L,M
  |
INPUT P
  |
Make the following computations
        A/6*P, B/6*P, C/6*P
        D/6*P, E/6*P, etc.
```

Outputting the Data

The final step in writing a program is outputting the data. The statements we created for calculating the amount of each ingredient were PRINT statements which will print out numbers. This format is inadequate because it will be difficult to keep track of what each number represents. We can solve this problem by simply adding string literals to our PRINT statements:

```
40 PRINT A/6*P;   "CUP OIL"
50 PRINT B/6*P;   "ONIONS"
60 PRINT C/6*P;   "CUP PARSLEY"
70 PRINT D/6*P;   "TS OREGANO"
80 PRINT E/6*P;   "LB BEEF"
and so forth
```

As you can see, string literals are a very handy labeling device. Our results will now be much easier to read and keep track of.

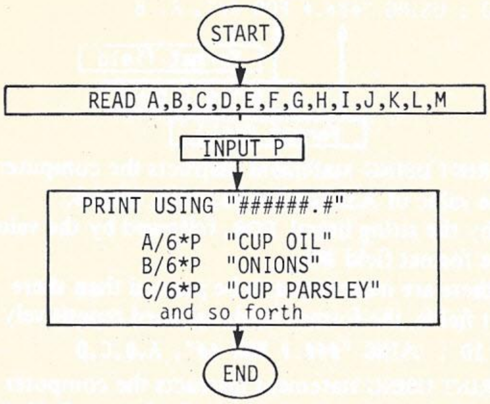
When you run the loans program in Lesson Two, you often get results that have to be rounded off. For example, one of our answers was, MONTHLY PAYMENTS ARE 339.375, which we rounded to 339.38. By substituting PRINT USING statements for the PRINT statements in the output part of our program,we can actually define the FORMAT we want our answers in.

For example:

```
40 PRINT USING "######.# ",A/6*P, "CUP OIL"
```

This PRINT USING statement allows for one digit following the decimal point. Whether the second digit is rounded off or simply dropped depends upon the BASIC you are using. In the case of our spaghetti program, this distinction is not important.

We can now complete our flow diagram:



The Program

Using the flow diagram as a guide, you can write a program to solve our "spaghetti" problem. Before you examine the program listing below, try writing it. The program:

```
NEW
10 READ A,B,C,D,E,F,G,H,I,J,K,L,M
20 DATA .25,1,.25,1,1,1.25,6,.75,1,1,1.5,.25,2
30 INPUT P
40 PRINT USING "######.#" ,A/6*P, "CUP OIL"
50 PRINT USING "######.#" ,B/6*P, "ONIONS"
60 PRINT USING "######.#" ,C/6*P, "CUP PARSLEY"
70 PRINT USING "######.#" ,D/6*P, "TS OREGANO"
80 PRINT USING "######.#" ,E/6*P, "LB BEEF"
90 PRINT USING "######.#" ,F/6*P, "LB TOMATOES"
100 PRINT USING "######.#" ,G/6*P, "OZ PASTE"
110 PRINT USING "######.#" ,H/6*P, "CUP WATER"
120 PRINT USING "######.#" ,I/6*P, "BAY LEAF"
130 PRINT USING "######.#" ,J/6*P, "GAR CLOVES"
140 PRINT USING "######.#" ,K/6*P, "TS SALT"
150 PRINT USING "######.#" ,L/6*P, "TS PEPPER"
160 PRINT USING "######.#" ,M/6*P, "LB SPAGHETTI"
170 END
```

Editing the Program

If you enter this BASIC program into a computer and then enter the command RUN, the computer will answer with a question mark (?). This question mark assumes that you know what it is the computer is asking you. Why not add a PRINT statement to our program to act as a label for the INPUT statement? All we have to remember is to give the PRINT statement a line number (see Lesson One) that puts it in sequence **before** the INPUT statement:

25 PRINT "NUMBER OF PEOPLE BEING SERVED"

line number

This time when you enter the RUN command, the computer will answer: NUMBER OF PEOPLE BEING SERVED? If you enter the number 25, the computer will give you the following results:

```
 1.0 CUP OIL
 4.2 ONIONS
 1.0 CUP PARSLEY
 4.2 TS OREGANO
 4.2 LB BEEF
 5.2 LB TOMATOES
25.0 OZ PASTE
 3.1 CUP WATER
 4.2 BAY LEAF
 4.2 GAR CLOVES
 6.3 TS SALT
```

```
1.0 TS PEPPER
8.3 LB SPAGHETTI
```

You can further edit the program to leave personal messages. These messages should come at the end of the program. For example:

```
163 PRINT "DON'T FORGET THE GARLIC BREAD."
167 PRINT "FOR A GOOD ITAL DRESSING SEE ITALD."
```

After printing out the answers, the computer will now remind you about the garlic bread (last time you forgot, you boob!) and tell you the file name of a good Italian dressing recipe stored on your recipe disk (see Lesson Two).

More on Print Using

All of the techniques used to solve this spaghetti problem, with the exception of PRINT USING, were techniques you learned in Lessons One and Two.

In the spaghetti program there are several PRINT USING statements. The first of these is:

```
40 PRINT USING "######.# ",A/6*P,"CUP OIL"
```

commas between items

space

format field

This PRINT USING statement instructs the computer to PRINT (or display) the result of the computation, A/6*P, within the limits of the format field "######.# ". A # in a format field represents any digit (0-9). The space in the format field is to allow a space between A/6*P and the string literal "CUP OIL" when the results are displayed.

A FORMAT FIELD may be contained in a FORMAT STRING with string literals, numbers and/or other format fields. An example of a format string is:

format string

```
10 ; USING "###.# FOR ##", A, B
```

format field

string literal

format field

This PRINT USING statement instructs the computer to display the value of A in the format field, ###.#, followed by the string literal FOR, followed by the value of B in the format field ##.

When there are more items to be printed than there are format fields, the format fields are used repetitively.

```
10 ; USING "###.# FOR ##", A,B,C,D
```

This PRINT USING statement instructs the computer to first print (display) the value of A in the format field ###.#, followed by the string literal FOR, followed by the value of B in the format field ##. It then instruct the computer to print the value of C in the format field ###.#, followed by FOR, followed by the value of D in the format field ##.

In the lessons that follow we will discuss new techniques for improving our Spaghetti program. Suppose you were the owner of a spaghetti restaurant. You need to keep better track of your inventory. You serve 25 different spaghetti sauces with eleven forms of pasta and six cheeses. Your business is surprisingly complex. Using the spaghetti program in this lesson as a base, we'll look for unique and powerful uses of our personal computers in dealing with such complexity both at home and at work. If you'd like to help, we'll pay $25 for the best suggestions. Write: Spaghetti BASIC, 401 Louisiana SE, Albuquerque, NM 87108.

Making the Transition from a Big System to a Micro or How to Get Along Without a Megabyte of Memory

by Jim Woodward

At the current stage of the market for personal computers, most owners and potential owners will have computer experience from larger systems. Many will have had large university time-shared machines at their disposal. Going from such an environment to the more limited one of a personal computer will be easier if the potential owner is prepared for it. This step has two parts: the first, known as benchmarking, is the art of getting the most for your money; the second, optimization, is the search for more speed or memory or whatever else you are shortest of.

Benchmarking is simply the measurement of a computer's speed, or thruput, in the jargon. The catch, particularly in large systems, comes with defining speed. A computer is a complex collection of input/output, central processing, other hardware and software. To say that Computer A is faster than Computer B requires defining the task. In selling large computers much of the sales effort may go into defining tasks in such a way that they favor a particular machine. Luckily, in microcomputers the complexity is reduced and simple benchmarking is both possible and relevant. Assume that your orientation is toward Basic and that you expect to run reasonably complex programs (whether Space War or a Monte Carlo business forecast model is immaterial). If you go to a couple of dealers and run even the simplest of Basic loops, you may be in for a big surprise:

```
10 FOR I = 1 TO 1000
20 LET X = X + 1
30 NEXT I
40 PRINT X
50 END
```

On two popular 8080 systems this simple loop took two seconds and 30 seeonds, respectively, from the push of "RUN cr" to the output of X. The *faster* system was operating with a complex disk-based Basic with a high overhead, while the slower system was a simple, Basic only, cassette-loaded "efficient" one. No doubt this 10 to 1 difference will change, but here and now you have a clear choice on speed and a clue to the manufacturer's orientation toward his product.

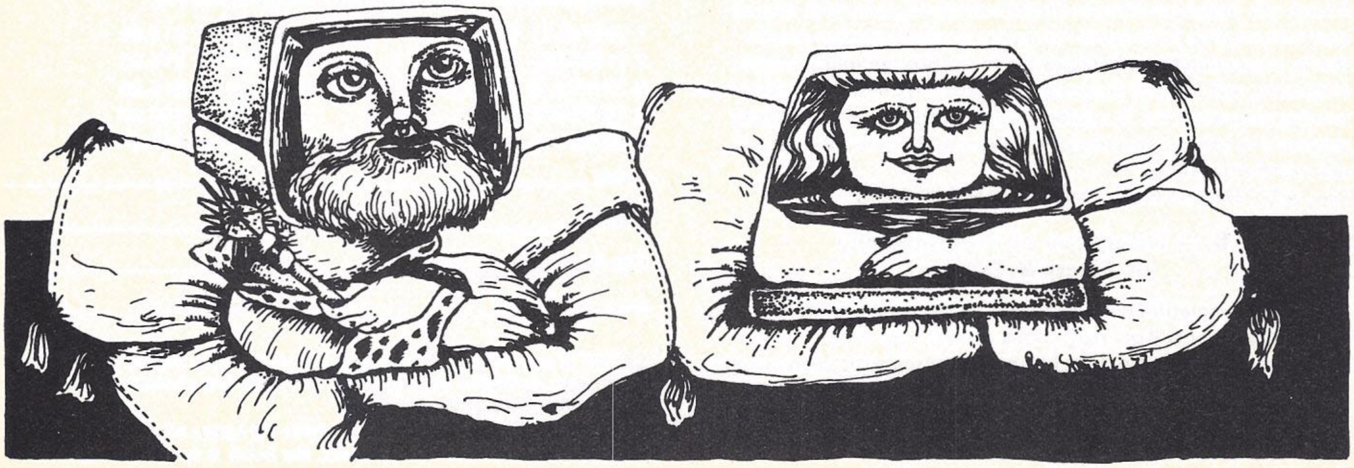
Follow this simple test with others, weighting them in directions which interest you. Try:

```
25 LET X = 5*X
26 LET X = X/5
```

inserted in the routine above. Both multiplication and division are fairly complex software in a micro and will be important in most applications. Will you use the library functions often? Try 1000 calls to the SINe routine for speed and check for accuracy against a 15-place table (your library will have Abramowitz and Stegun, *Handbook of Mathematical Functions).* Try the RaNDom numbers:

```
10 FOR I = TO 1000
20 LET X = RND (1)
30 LET X = 1 + INT (10*X)
40 LET A (X) = A(X) + 1
50 NEXT I
60 print the 10 values of A in some pleasing format
```

The statistician can provide tests like Chi-square to look at the randomness in a sophisticated way, but in broad terms, if one of the counts is less than 90 or more than 110, the generator *might* be less than acceptably random. Continue, looking at anything else that interests you. The dealer will see you looking at your watch and will know what you're doing, but it shouldn't bother him, unless he knows his com-



puter will come out on the short end.

Don't, incidentally, expect the sheer blinding speed of a large system; larger systems have both shorter cycle times and larger instruction sets. In addition, they typically work on longer strings of data at once, for a triple barreled increase in speed. If you're planning on inverting large matrices or running numerical problems of unusual complexity, consider spending more on a larger system or on time-sharing.

Having looked at speed, now look at size. There are 2K and 4K and 5K and 8K and 12 K and 16K Basics available, but most of these numbers tell you little. Some of the software actually takes the full "xK" for Basic, leaving nothing for your program, while others allow a good sized program in the advertised space. Rather than take the printed specs verbatim, test a little. The quickest way to use up space is to DIMension large arrays. Try running:

```
10 DIM A(500)
20 FOR I = 1 TO 500
30 LET A(I) = A(I) + !
40 NEXT I
99 END
```

Note that the 1 in statement 30 is an integer; both size and timing *may* vary for different non-integral values here.

Now run it again and again, changing the 500 in both places until you know exactly the permitted size small enough to run without getting an error message such as TOO BIG AT 10. You now have a rough measure of the capacity for user program. Once you have established this limit exactly, add 5 or 10 lines of program such as some of the tests from above. The largest possible DIMension will be reduced; simple proportions will tell you how many more lines of program you could add if you eliminated the DIMension altogether. I doubt if you'll be happy without capacity for 100 lines or so, which might be 2000 bytes at 20 characters average per line.

We looked first at speed, in detail, and then at capacity, in a much more rough and ready fashion. There is good reason for this: speed must be lived with while capacity is curable. Up to the 64K address limit of most micros you can add capacity for 400 lines of Basic (8K) for about $200 today. Five hundred lines of Basic is a *big* program. Speed is harder to fix. Improved software may help, but don't spend a lot of money on a new superfast version without trying it out. More hardware may help. In designing a machine for sale, it is very tempting to let software handle a lot of tasks that could be handled by hardware. After all, once the software is written, it costs next to nothing to supply it with the machine. More sophisticated hardware, on the other hand, is more expensive to supply, service and document. There is at least one supplier of a board which implements floating point arithmetic in very fast hardware, claiming, not unreasonably, a 50 to 1 speed increase. There are also many different ways to handle output devices, some of them much faster for the CPU but more expensive in hardware.

Having benchmarked the various possibilities and bought the best of them, how do you make the most of your limited resources? This step, more art than science, is known in the trade as optimization. It may be memory size, speed, input-output or plain cost, but any resource limiting your programming must be conserved. Inveitably this involves compromise, and each of the tricks below has an offsetting disadvantage in another area.

To reduce memory requirements: Use short line numbers; eliminate blanks wherever possible; use multiple statements per line if permitted; eliminate REMarks; keep printed messages short; reuse variable names wherever possible; and avoid most everything below.

To increase speed: Gosub sparingly; minimize jumping from one section of a program to another (works on some machines); experiment to see whether FOR-NEXT is faster or slower than IF-THEN for simple loops; write;

```
LET A = B + B + B + B
LET A = 4*B
```

(multiplication is usually *much* slower than addition so that this technique can work for quite large numbers – experiment!); similarly

`100 LET A = B*B*B*B` may be faster than
`100 LET A = B 4` and it may even pay to write
`100 LET A = B*B`
`120 LET A = A*A` on some machines;

allow the user to select the level of detail desired in printed messages ("PLEASE PROVIDE THE MONTHLY PAYMENT AMOUNT?" vs "PAY?"); experiment with simple code versus complex expressions;

```
100 LET X = SQR ( B 2 - 4*A*C)
```

may take longer than

```
100 LET E = B*B
110 LET F = A*C
120 LET F = F + F + F + F
130 LET D = E - F
140 LET X = SQR(D)
```

Of course, this example ignores the question of whether the discriminant (D) is negative, a condition you should probably test in any event.

To forestall screams of anguish complaining that most of the methods above make reading a program difficult, I admit it point blank. Cramming a program into minimum space and time will not endear you to future readers, including yourself, if you do not keep careful notes to accompany the program wherever it goes.

Where space and speed permit: Use REMarks plentifully; increment line numbers by 10 to leave space for future changes; write only one statement per line; avoid special bells and whistles peculiar to your machine's version of Basic. Leaving aside the possibility of selling a program, using all the little tricks will ruin you if you buy a new CPU board, or add hardware incompatible with the special tricks. Reuse only index variable names (many old hands instinctively use I, J, and K for index, or counting, variables throughout the program and L, M, and N for the end points of loops:

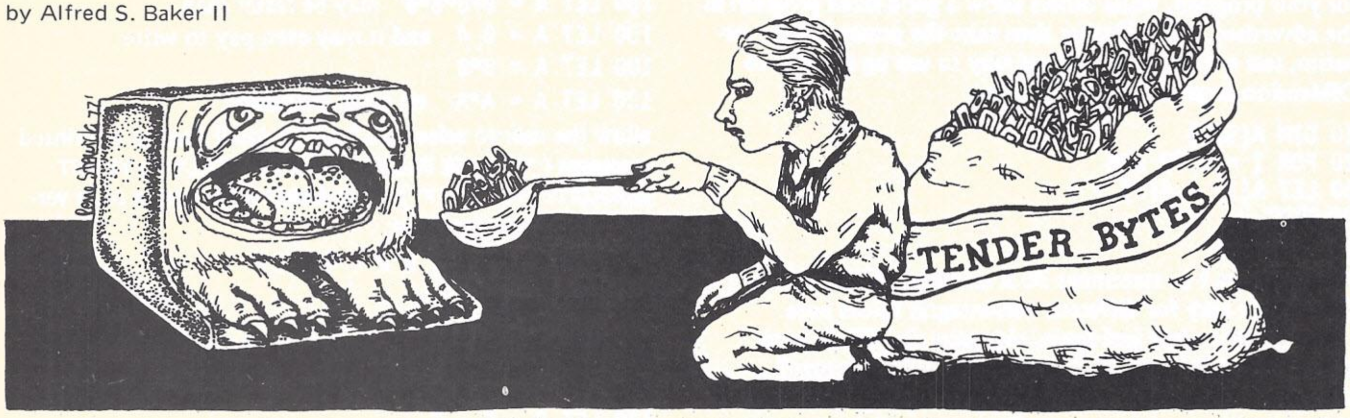
```
100 FOR I = I TO M
```

This habit developed from early versions of Fortran which required this form of name. The habit is a good one; in reading your programs you will know that those six letters designate counting variables.). Avoid the letter 0 as a variable name. Whether your machine puts a slash across the numeral 0 or capital O, confusion is inevitable and avoidance is best. Use the rest of the alphabet for calculation variables. Unless tight for space do not reuse variable names. It can cause no end of confusion if you use "C" in the first few lines for one thing and then, not needing it any more for that, reuse the same name for something else later on.

Throughout this article the theme has been experimentation. Besides being fun, it will tell you things about the machine that may not be published or even known. While blind concern for speed and memory may waste time and money in the long run, knowing what you can do if needed is bound to improve your work; doing it on the best machine for your needs will further increase your enjoyment.

A Simple Technique For Making The Right Computer Buy

by Alfred S. Baker II



Don't waste your money. While owning a computer is now financially feasible for almost everyone, it is still not an inexpensive purchase, just as good stereo system, large appliance, car or home is not. Making the right decision can mean the difference between long-time satisfaction and disappointment. You are a manager. Whether you're managing a multinational corporation, your pocket book, your home or your lemonade stand, one of the principles of successful management is rational decision analysis*: how to make the best decision given one's wants, requirements and objectives. I will demonstrate how you can use a straight forward management technique to purchase the computer of your dreams.

Table 1 contains an outline of the basic decision making process. First, what do you really want in a computer? What must it have and what would you like? Remember, your needs are not my needs or the next guy's needs. Example 1 is is a possible list of such needs. You can add to the list, subtract from it or throw it out and start all over. The important thing is to *make* it! Think about it. Read the literature describing various offerings. This magazine and others carry reviews of the various products on the market. The manufacturers pay for ads and have brochures available for the asking.

As you read these articles or ads, add to the list when particular features catch your eye. Also, think about where you are going. What applications would you eventually like to use the computer for? Are floppy disks or other types of mass storage needed and does this or that computer support them? What other kinds of external devices do you have in mind? What are your immediate and eventual memory requirements? What are your requirements for read-only memory? Rememeber that most of the computers on the hobby market don't do very much by themselves. Given your immediate and long-range desires, how much is it going to cost to add the rest? Computers also eat software. What kind of

Example 1 Preliminary list of features, requirements, and objectives

1. How easy is it to assemble?
2. Is it a kit?
3. What is its basic price?
4. How much RAM is supplied? How much does more cost?
5. How much ROM is supplied? How much does more cost? What is on it?
6. What software is available on ROM for this computer?
7. Is a Monitor supplied? (On ROM?) Is one available? How much?
8. Is an Assembler supplied? (On ROM?) Is one available? How much?
9. Is an Editor supplied? (On ROM?) Is one available? How much?
10. Is BASIC supplied? (On ROM?) Is one available? How
11. How many vendors supply Bus compatible equipment?
12. How many vendors supply compatible software?
13. How much software is available in the public domain?
14. How many extra slots are there on the motherboard?
15. How fast is the computer for the kind of use it is to be put?
16. Is the machine instruction set to your liking?
17. What is the range of external devices which are supported by bus compativle interfaces? Are the specific devices you have in mind supported?
18. How many alternative suppliers for these interfaces are there?
19. Does it support inexpensive mass storage (eg. cassette)?
20. Does it require the use of a teletype compatible device?
21. Does it have a large enough power supply?
22. How many people close by or in the club own it?
23. What is the reputation of the manufacturer?
24. What is the prestige of owning it?

software will you want or need?

All these questions should be reflected in your list. Failure to answer any one of them in your requirements list should be intentional. If you really don't consider a question important. Fine. Just be sure the omission is not an accident.

Try to keep together all the articles, ads, brochures and other materials you have collected. As you eventually begin to rate the various products, you will need this information to grade them properly. Having the literature in one place for review will give you a chance to organize your thoughts easily. Some features will seem so unimportant that they will attain the status of a veto vote. Your computer *must* have this or that option, or it must be able to do such and such, or it cannot cost over X dollars.

Once you've prepared the list, place all those items which are MUSTS on one sheet of paper and all the rest, which become your WANTS, down the left edge of another. Table 2 is an example of such a MUSTs list and Table 3 is an example of a WANTs list.

Table 1 The Decision Making Process

I. Generate a list of features, objectives and requirements. This is the most time consuming and important part of the process.
II. Separate out the absolute MUSTS and weigh the remaining WANTS.
III. Determine the alternatives (choices).
IV. Eliminate any alternatives which do not satisfy *all* MUSTS.
V. Grade each alternative against each WANT (0 to 10) and multiply by the weight to produce a rating.
VI. Sum the ratings for each alternative to give a total satisfaction value or score.
VII. Take the 2 or 3 best choices, by score and disadvantages before you choose.

Table 2 MUSTS list

1. Must be a kit.
2. Must not require a teletype compatible device for start up.
3. Including TV, cassette, and keyboard interfaces; keyboard; and 8K of memory; it must cost less than $1500.

Table 3 WANTS

WANT	Wt.	Brand X Mod 1	Brand Y Mod 1	Brand Z Mod 4	Brand A Mod 2	Brand B Mod 1	Brand C Mod 2	Etc.
Ease of assembly	5	7 35	5 25	FAILS	FAILS	FAILS	5 25	
RAM memory supplied	3	0 0	0 0	MUST:	MUST:	MUST:	10 30	
ROM memory supplied	1	0 0	0 0	Req.	Req.	Not	5 5	
Assembler	10	8 80	8 80	TTY	TTY	a Kit	8 80	
Editor	9	8 72	8 72				8 72	
Monitor	7	8 56	8 56				8 56	
BASIC	5	8 40	10 50				8 40	
Compatible BUS vendors	10	10 100	10 100				0 0	
Compatible software vendors	5	10 50	10 50				5 25	
Cassette Interface	10	8 80	8 80				0 0	
Big power supply	10	10 100	8 80				0 0	
Club member experience	5	8 40	3 15				1 5	
Mother-board slots	5	10 50	8 40				1 5	
Low price	5	5 25	3 15				8 40	
Prestige	5	5 25	10 50				0 0	
Total Score		753	713				383	

Disadvantages:

BRAND X MOD 1 (753)

1. No company sponsored user group.
2. Seems to lack hobbist commitment.

Disadvantages:

BRAND Y MOD 1 (713)

1. Delivery time.
2. Very new product (little field experience).

Now you are ready to weigh your WANTs. Using Table 2 as a guide (remember, your list of WANTs will be different), write a weight down for each WANT. The most important WANT (or WANTs) should be given a weight of 10 and all other WANTs should be judged in proportion to these: half as important=5, two-thirds= 7, one-fourth= 2 or 3, almost as important= 9, and so on. Finally, list the alternatives (in this case computers) across the top of the WANTs sheet. The way the market is expanding, you may have to tape one or more sheets of paper to this one! Eliminate those alternatives which do not satisfy *all* MUSTs from the MUSTs sheet.

You are now ready to rate and score all of the alternatives against your WANTs. Under each alternative make two columns. In the first column grade how well each alternative satisfies each WANT (0 to 10) and in the second column place the result of this grade multiplied by the weight of the WANT. This result is the rating. When this has been done for all of the alternatives, sum each rating to produce its score.

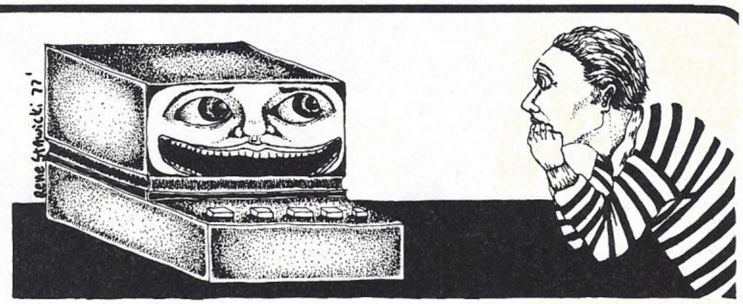
Now, choose the best two or three alternatives by picking those with the highest scores. Scores differing by no more than 5% should be considered essentially the same. At the bottom of the sheet of paper list these best alternatives with their scores. Under each one, list its disadvantages. Here again you may want to spend some time. Go back over the material you have collected and try to pick apart these choices. This analysis can be hard, especially if you really like them. Last of all, with these choices and their scores and disadvantages before you, choose your computer.

This procedure isn't magic and it doesn't make your choice for you. It simply gives you a way to organize your thinking and keep track of your alternatives and needs. Also, depending on the amount of money you are thinking of spending or the time available, you may want to do it several times. Review your MUSTs. Are they *really* MUSTs? Have they kicked out any alternatives that you really want to consider? Review your WANTs. Have you thought of them all? Are you being honest with yourself? Are they weighted properly? Have you thought of all your alternatives or at least the important ones? And finally, have you *really* thought about their disadvantages?

These simple techniques force decisions for managers on a rational basis. Perhaps they will help you spend your money wisely in seeking the computer system that suits you best.

RX For Hardware Phobia

IF YOU'RE TIRED AND LISTLESS BECAUSE YOU CAN'T AFFORD THE FULLY ASSEMBLED SYSTEM OF YOUR DREAMS AND YOU'RE AFRAID OF DO-IT-YOURSELF, THIS LITTLE PRESCRIPTION MAY BUCK YOU UP.



by Jim Woodward

Are you a computer user who has never been closer to one than a Teletype keyboard? Would you like your very own personal computer, but fear of all that hardware scares you off? Do you have a vague idea what resistors and capacitors do, but can't remember which end of a diode is which? Read on.

I don't propose to turn you into a hardware designer in a short article or even tell you how to go about it; I do propose to convince you that an investment of a few dollars and a little time will pay off in a better understanding of what goes on inside that micro-box. Half a dozen various building blocks, called chips or DIPs (for Dual Inline Package, a description of the physical arrangement), can be strung together to do a variety of things. To me, most amazing about them is their great variety; the fact that there are hundreds of different building blocks available is more impressive than that some of them have the immense complexity of microcomputers. The 7400 series (74 is an arbitrary number that the industry uses as a prefix for the most common and useful type) of TTL chips (Transistor-Transistor-Logic, a description of the internal workings which you may ignore entirely) is almost immune to damage from a reasonably careful person. A 7400 chip can be touched, submerged (briefly), shorted, dropped, soldered and unsoldered without much chance of damage (not so with some other logic series such as MOS). This immunity to abuse and low cost, down to eight for $1 for some types, makes them ideal for a beginning experimenter. While single chip devices are available for most common uses such as clocks, calculators and so forth, there is great fun in designing your own from scratch work less complex TTL chips.

The best easily available introduction to the subject is in *Scientific American,* May 1973, page 108. While this article is simplistic almost to a fault and omits some precautions that will make the experienced designer cringe, it is readily available from your local library and can be read by the beginner. If you want more, by all means buy Don Lancaster's *TTL Cookbook,* which covers all of the common chips and many sophisticated methods.

The most expensive single item you will want is a solderless plug-in breadboard, designed for TTL DIPs. Radio Shack has them at $9.95 as do most computer stores. By eliminating soldering, these make it possible to wire a complex circuit fast and reuse everything as you move on to the next experiment. Start perhaps with a clock; although they are available on single chips, you will learn a great deal in making a display count from 1 to 60, once a minute. Try making music; an 8-ohm speaker can easily be driven between the output of a chip and the positive side of the power supply by putting two 100-ohm resistor in series with the speaker (I am deliberately vague on the resistor size; TTL doesn't care much what you do to it, but do use the TTL chip for the low side of the circuit). You can easily get different tones by putting different dividers in the circuit; going from this to a full electronic music maker is conceptually easy, although making the keyboard and the enclosure may be difficult.

You won't qualify to go to work as a computer designer, but you will find that TTL logic is so cheap and easy that you won't be afraid to tackle larger projects. Suddenly the computer with all of its complexities becomes comprehensible, albeit one step at a time. The thought of designing your own "perfect" display interface or computer driven music maker is not only possible but reasonable.


WIRE WRAPPING TOOL
For AWG 30, .025" (0,63mm) sq. post, "MODIFIED" wrap, positive indexing, anti-overwrapping device
OK MACHINE & TOOL CORPORATION
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091
NEW
HOBBY-WRAP
Model BW-630
Battery wire wrapping tool
$34.95 ONLY
(batteries not included)
COMPLETE WITH BIT AND SLEEVE
NEW
HOBBY-WRAP
Model BW-630
Battery wire wrapping tool
$34.95 ONLY
(batteries not included)
COMPLETE WITH BIT AND SLEEVE
WIRE WRAPPING TOOL
For AWG 30, .025" (0,63mm) sq. post, "MODIFIED" wrap, positive indexing, anti-overwrapping device.
OK MACHINE & TOOL CORPORATION
3455 Conner St., Bronx, N.Y.10475 / (212) 994-6600 / Telex 125091

CIRCLE 34

The Small Computer

Twenty-five years ago a computer as powerful as the new Processor Technology Sol-20 priced out at a cool million.

Now for only $995 in kit form or $1495 fully assembled and tested you can have your own small computer with perhaps even more power. It comes in a package about the size of a typewriter. And there's nothing like it on the market today. Not from IBM, Burroughs, DEC, HP or anybody else!

It fills a new role

If you're an engineer, scientist or businessman, the Sol-20 can help you solve many or all of your design problems, help you quantify research, and handle the books too. For not much more than the price of a good calculator, you can have high level computer power.

Use it in the office, lab, plant or home

Sol-20 is a smart terminal for distributed processing. Sol-20 is a stand alone computer for data collection, handling and analysis. Sol-20 is a text editor. In fact, Sol-20 is the key element of a full fledged computer system including hardware, software and peripheral gear. It's a computer system with a keyboard, extra memory, I/O interfaces, factory backup, service notes, users group.

It's a computer you can take home after hours to play or create sophisticated games, do your personal books and taxes, and a whole host of other tasks.

Those of you who are familiar with small computers will recognize what an advance the Sol-20 is.

Sol-20 offers all these features as standard:

8080 microprocessor — 1024 character video display circuitry — control PROM memory — 1024 words of static low-power RAM — 1024 words of preprogrammed PROM — built-in cassette interface capable of controlling two recorders at 1200 bits per second — both parallel and serial standardized interface connectors — a complete power supply including ultra quiet fan — a beautiful case with solid walnut sides — software which includes a preprogrammed PROM personality module and a data cassette with BASIC-5 language plus two sophisticated computer video games — the ability to work with all S-100 bus products.

Full expansion capability

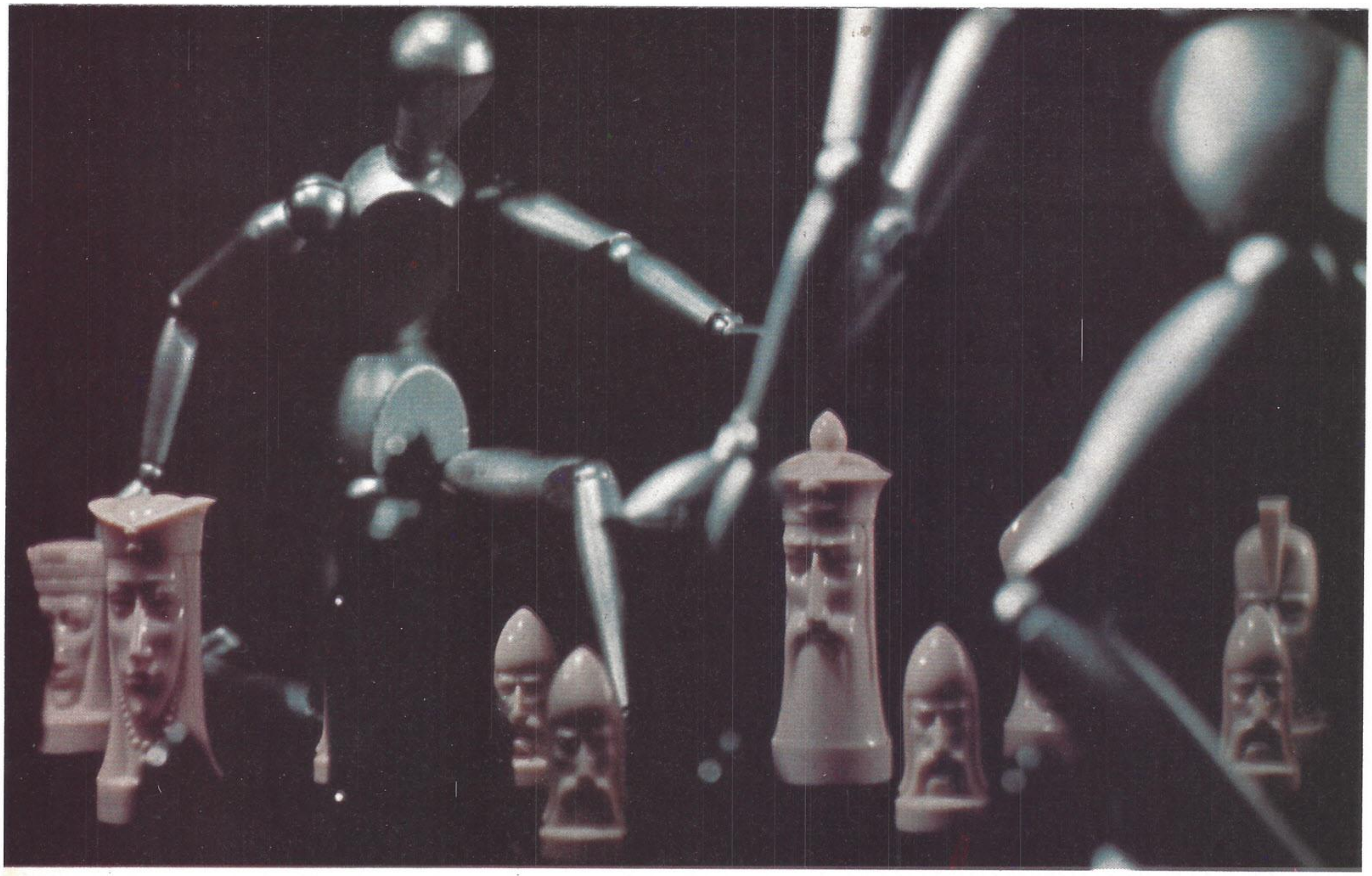
Tailor the Sol-20 system to your applications with our complete line of peripheral products. These include the video monitor, audio cassette and digital tape systems, dual floppy disc system, expansion memories, and interfaces.

Write for our new 22 page catalog. Get all the details.

Processor Technology, Box L, 6200 Hollis St., Emeryville, CA 94608. (415) 652-8080.

Processor Technology Corporation

CIRCLE 35



A CHESS PIECE

Yes, of course chess enthusiasts are putting personal computers to work in pursuit of the game. Here's a quick look at the field for the non-fanatical chess player.

by David Galef

In the past five years, there has been an increasing amount of interest in merging one of the world's oldest games with the artificial brains of modern technology: programming a computer to play chess. While many specialized articles have been written on the subject, the focus is often on abstruse technical problems associated with the process; the reader comes away with the feeling that programming a computer to deal with the abstractions of chess is comparable to teaching a monkey to read Shakespeare. Not true! Steady progress is being made continually: smaller and more compact systems are playing better and better chess.

In 1972, while Fischer was making news in Iceland, the United States Computer Chess Championship took place in Boston. The winning entrant was a Control Data Corporation 6400 computer with a program called CHESS 3.5. The moves were relayed by teletype from Northwestern University. Since then, newer and more sophisticated programs and machines have moved up through the ranks. In October of 1975, Northwestern came through again with first place in the sixth annual championship tournament sponsored by the Association for Computing Machinery. The new program was CHESS 4.4—a not-so-distant relative of the earlier winner.

The principles of computer chess are not hard to explain, though the actual programming requires a good deal of skill – and fortitude. Leaving out the technical work involved (and incidentally, BASIC is not a good language for this type of programming), the attributes of a good chess-playing system are the traits of any good flesh and blood player.

The first thing to teach the computer is to look ahead. This is done by the flow chart or branching system, wherein one move means six possible moves for the opponent, each of the six moves leads to six replies, and so on. High-speed computers handle this process particularly well, analyzing a large number of possible moves in a few minutes. Another important feature in any chess program is what chess players call "book knowledge" – the memorization of a number of chess openings so that the opening of the game, if played correctly, can be over and done with in half a minute. Since a computer can easily store quite a few openings in its memory, this is one of the most atractive and promising features of chess-playing computers. No one, however, has yet attempted to feed MCO (Modern Chess Openings, the chess player's bible) into a system: the number of variations and sub-variations is enormous.

This, of course, brings us to the last practical "must" in the program: even though the computer is an extraordinarily fast machine, some way of teaching the computer to limit the possible moves it considers must be found. The reason for this becomes painfully obvious to anyone who attempts to write a program **without** this provision – there are over 169, 518, 829, 100, 544, 000, 000, 000, 000, 000, ways to play the first ten moves of a chess game! By restricting the focus of moves that the computer

considers, the programmer can greatly increase the depth of analysis in each choice the computer explores, since the computer will then have much more time to devote to a particular line of play. Yes, time *is* a factor, especially when some systems can take over an hour a move.

It's not so difficult to say what you want the program to achieve: you want the computer to play decent chess, using the guidelines already outlined. The problems arise in the actual programming of the computer, when you find it difficult to translate what you want the computer to do into a workable program – chess is a complex game. Then, too, there are some peculiar bugs that seem to plague chess programs in particular.

The first bug that appears in a chess-playing program will be found in most complex programs: the computer is given such a complicated path to follow that a foul-up will occasionally occur. Either the computer will make an outright blunder in play or miss a step somewhere. . .this, of course, is a law every programmer knows: the more complex the set-up, the more things there are to go wrong.

Another problem that will arise is the actual type of playing the computer will produce – a slow, conservative style, lacking in dynamic, sacrificial play. Conversely, the computer will grab at its opponent's pieces, even when snatching an opponent's pawn means falling into a rather obvious trap. Unless you are prepared to run the program on some crazy random move basis, this foible is unavoidable. The rule which must be heavily stressed in a chess program is not to lose pieces, and this factor tends to outweigh other considerations, even when the opportunity for a sacrifice lies open and waiting, only two moves away. What programmers often use is a combination formula-definition called mobility and multiply it by a constant. If the numerical value of the piece the computer is risking is less than the mobility factor that will be attained, the computer might choose to make a sacrifice – a rare occurrence in computer chess.

Development in the field has had its share of woes, but there are many bright moments. Witness the game shown – White played by program, Black by a consultation team of two humans. The level of play represented is not very impressive: what *is* amazing is that the SCHACH MV5.6 program had no look-ahead capability! A good program would have a United States Chess Federation rating of about 1500; a person with a comparable rating would be known as a class C player.

The techniques keep improving, however. There are computers nowadays which benefit from games that play by learning from their mistakes – all as part of the program. With the computer's use of such techniques, even a few masters have succumbed once or twice to the mechanical Morphys. Not too long ago, there appeared an article in the New York *Post* advertising a minimicrocomputer to be marketed soon, at about $120, whose sole purpose is to play chess. The manufacturer, Cardinal Industries, Incroporated, claims the machine can operate on eight different levels of play, from a few seconds a move to a few hours, as the standard of play goes up. The programmer, David Lindsay, acknowledged that it was no Fischer but did say, "I must confess it beat me once at level three."

As a closing note: the British master David Levy has bet $2500 that he will still be able to beat any chess-playing computer by the year 1978. He claims he is still 100% sure of victory. . . but Mikhal Botvinnik, three-time winner of the World Chess Championship and a computer and electronics expert, as well, is reported to have told him, "I feel very sorry for your money."

White SCHACH MV5.6	Black Fischer/Schneider	White SCHACH MV5.6	Black Fischer/Schneider
1 N-QB3	P-Q4	13 Q-R6	N-B4
2 P-Q4	B-N5	14 BxN	NPxB
3 P-B3	B-B4	15 N-KN5	QxNch
4 P-K4	PxP	16 QxQch	K-R1
5 PxP	B-Q2	17 P-KN4	PxP
6 N-B3	N-QB3	18 QxP	P-B4
7 P-K5	P-K3	19 Q-R4	P-B5
8 B-KN5	B-K2	20 N-K4	P-B6
9 Q-Q2	P-KN3	21 N-N5	R-B2
10 B-Q3	P-N3	22 NxRch	K-N1
11 BxB	KNxB	23 Q-B6	P-B7
12 0-0-0	0-0	24 N-R6mate	

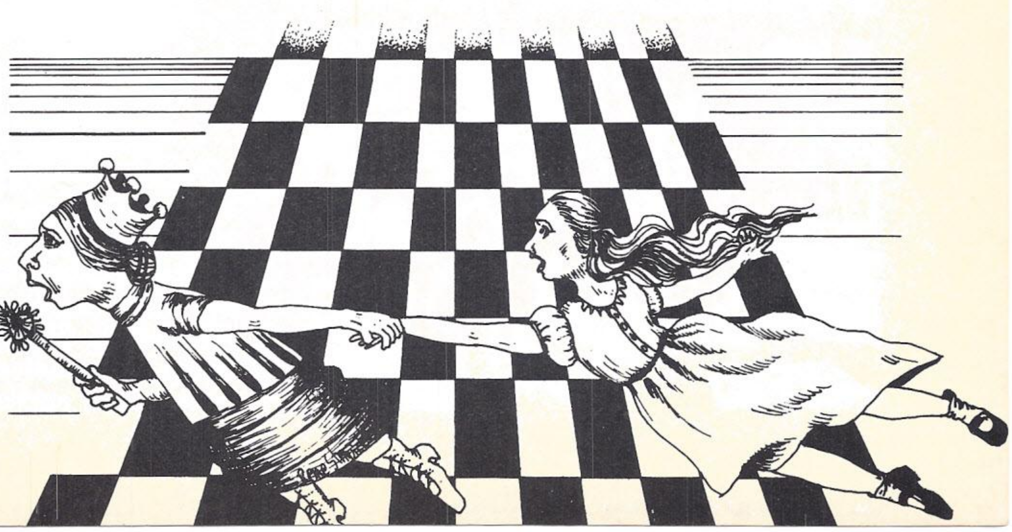
MICROCHESS

Unlikely as it seems, a tight little chess-playing software package is available for the KIM-1 6502 microprocessor system. The developer reports that the program and data occupy only 1100 of the 1152 bytes of available RAM. No additional memory is required and no peripherals are required; all moves are entered and displayed via the KIM keyboard and LED display.

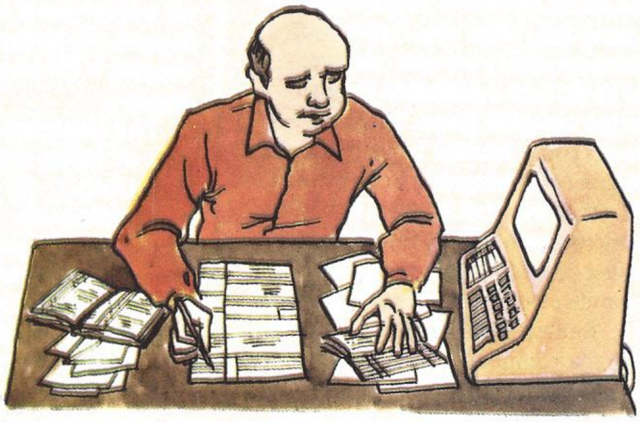
The program can be adjusted to one of three different levels of play requiring 3, 10 or 100 seconds for each computer move. "Although the level of play is probably below that of the average serious chess player, it is a good match for the average computer programmer."

The documentation supplied includes a Player's Manual with instructions for using the program, a Programmer's Manual with details of the program operation and suggestions for expansion and modification, and a complete annotated source listing.

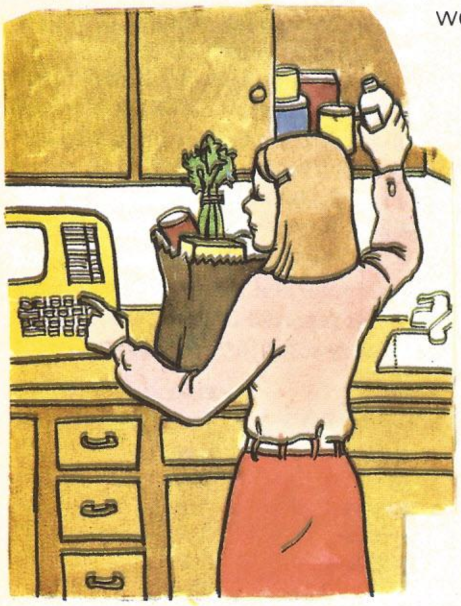
PERSONAL COMPUTING hasn't checked this out with Bobby Fischer, but the package sounds interesting at $10. Write Microchess, 1612-43 Thorncliffe Pk. Dr., Toronto, Ont. M4H 1J4, Canada.



PERSONAL COMPUTING is an exciting new magazine about revolutionary developments in personal computers. Designed around miniature IC chips, these computers are now being sold through retail stores at about the price of a home stereo system. Experts project that over 1,000,000 personal computers will be sold in the next five years.



The impact of personal computers will be enormous. It is the job of PERSONAL COMPUTING to make certain you know what computers are, how they work, and in what ways they are changing the world around you.

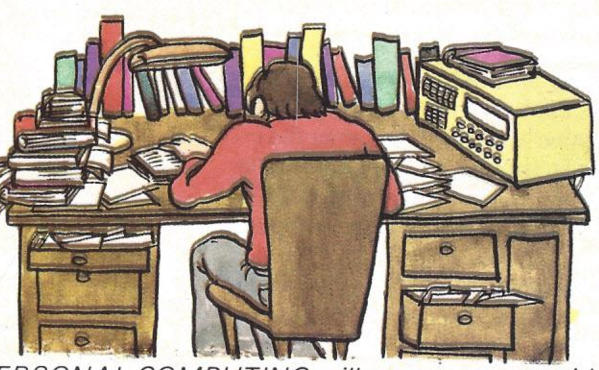


If you don't have your own computer by now, chances are there's one in your immediate future. Today's personal computers can do the following tasks:

Family and personal business management
Teaching aid for you and your children
Robot and industrial control
Inventory management
Accounting
Figure income taxes
Sophisticated text editing
Immediate access to files of information
Monitor and control energy consumption
Computer games including pong, chess and Startrek
Store and alphabetize addresses and phone numbers

and much more. New applications are appearing daily. Many of them are featured in PERSONAL COMPUTING.

PERSONAL COMPUTING wants to reassure you about the ease with which you can master your own computer. Each issue contains easy-to-follow, nontechnical articles on computer programming and computer jargon. These articles illustrate the use of personal computers at home, school, and work.



PERSONAL COMPUTING will serve as your guide into this exciting technology. We'll provide you with all the up-to-date information on new computer products, computer clubs, computer retail stores, and important personal computing events. PERSONAL COMPUTING will help you find the right computer system and the right software for your own, special application.

Each issue of PERSONAL COMPUTING contains the following:

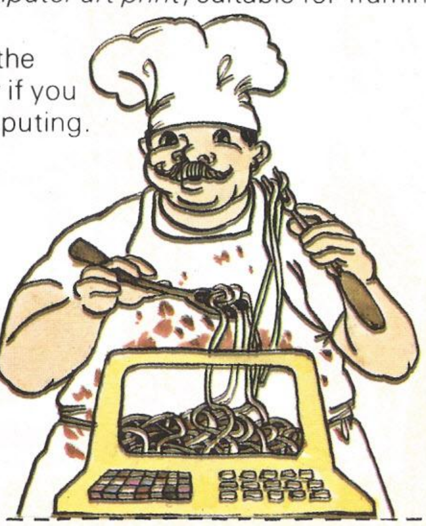
- **Lemonade Computer Service Company** section dedicated to the profitable uses of personal computers. We'll show you how to use computers in your own business and include free program listings to help you along.

- **Future Computing** narrative that speculates on the future developments and uses of personal computers. Each issue we invite a knowledgeable personal computing figure to write this column, thus giving you a wide spectrum of interesting and oftentimes controversial views.

- **Interview** with prominent person who may influence the growth and development of personal computing. Those already interviewed include Donn Parker, the nation's leading expert on computer crime, and Jules Bergman, science editor of ABC News.

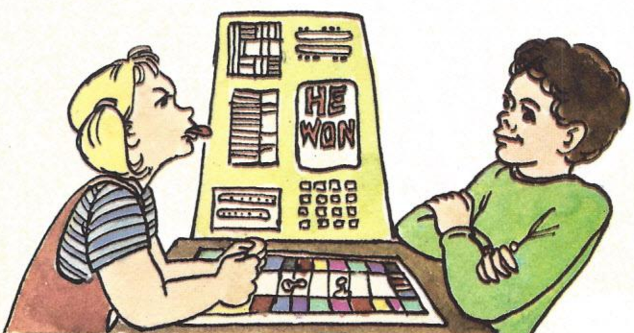
- *Free, full color computer art print*, suitable for framing.

- Special **Microtrek** section containing the technical information you'll need to know if you wish to get into the nuts and bolts of computing.



- **Spaghetti BASIC** programming course based on the proposition that learning to use a computer is as easy as learning how to make spaghetti.

- *Exciting computer games* including Hamarabi, Startrek and Backgammon are explained and free program listing provided.



In addition to these regular features, PERSONAL COMPUTING brings you news and stories from around the computing world. Our professional staff strives to bring you the best possible variety of articles, written in a fresh, lively style that you can easily understand.

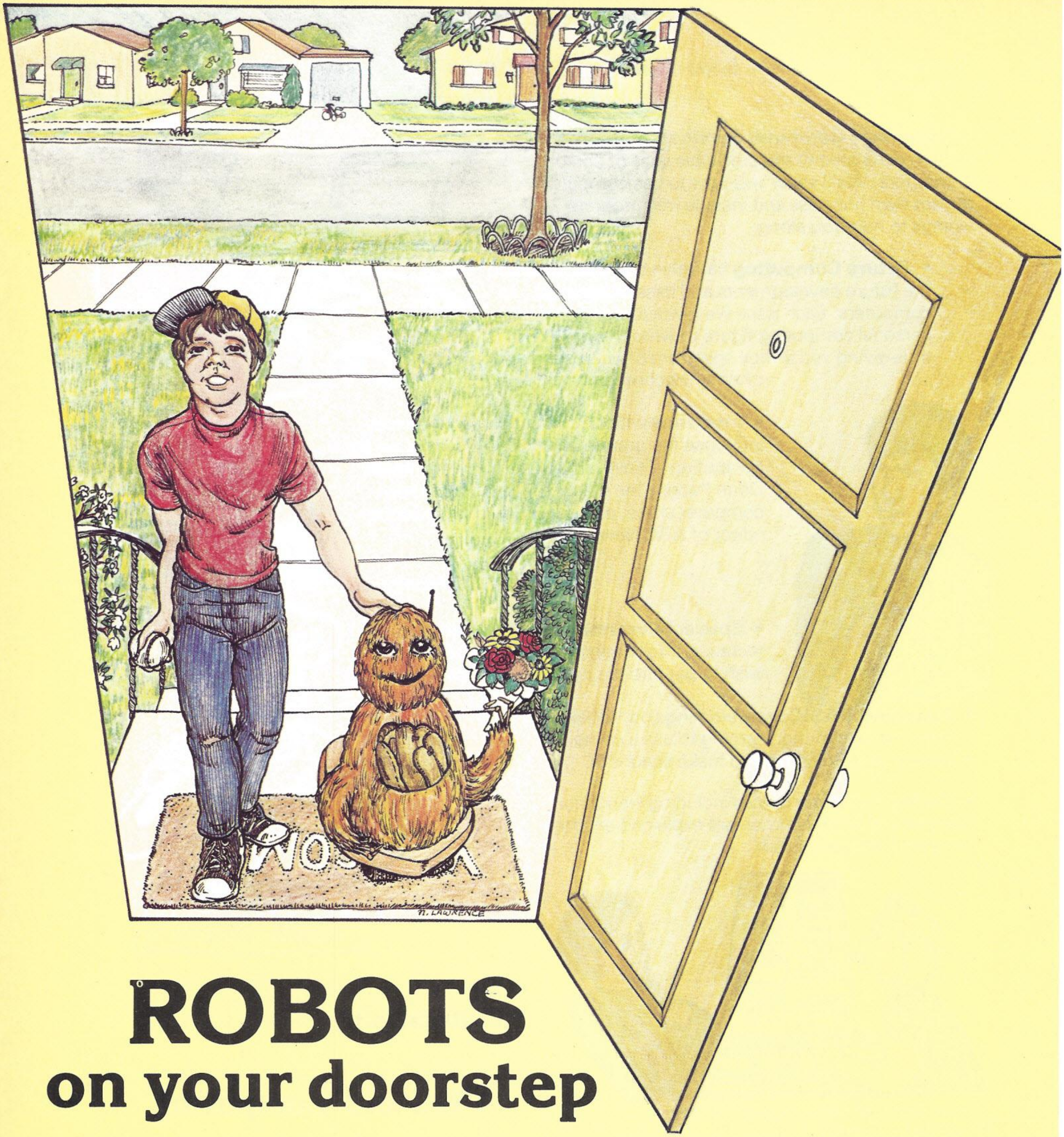
Published every other month, PERSONAL COMPUTING is an idea whose time has come. Subscribe now so you can keep up with this fascinating technology. You'll save $4 over the single-copy price.

To subscribe to

Personal Computing

Use convenient order form on

Reader Service Card



ROBOTS on your doorstep

A Personal Computing excerpt from a forthcoming book by Nels Winkless III and Iben Browning.

Have you always thought of robots as the product of a massive, expensive effort under the guidance of some super-agency like NASA, with its hoards of white-coated scientists? Times have changed, and robots are now more likely to be assembled on the amateurs kitchen table than in a giant laboratory.

FROM CHAPTER 1

Introduction

For many years now a few people have known that real robots are possible. A "real robot" is a thinking machine, not just a mechanical man programmed to react to predictable inputs, but a manufactured creature, probably self-reproducing, self-aware, interested in the world around it and concerned about matters it does not understand.

Robots have been acceptable in fiction, but the notion of their reality has been distressing to people who grow fearfully concerned about matters they do not understand. The few scientists working in the field of artificial intelligence have not always soothed the public with their diplomatic behavior, either.

As recently as 1973 the Associated Press distributed a story (without a by-line) that discussed research on artificial intelligence in language so plain that the Denver Post headlined the article: "THINKING" COMPUTERS SIMULATE HUMAN ACTS, STIMULATE FEARS. The

subheadline said TAKEOVER BY MACHINES?

The article pointed out that one well-known scientist had forsworn work on artificial intelligence on humanitarian grounds and was actively attempting to dissuade his students and colleagues from working in the field. The article also quoted some scientists who discounted any threat from robots (on the peculiar grounds that the things would not work all that well, anyway), but made much of one researcher's argument that "secret" research in the field should be forbidden. Only "open research carried on by the government" should be allowed.

The article did not explain what penalties were envisioned for those who asked questions and performed experiments not authorized by the government. No explanation was given of the mechanism by which it would be decided what would be authorized and what wouldn't. Presumably a committee of experts (including, ideally, the scientists who were demanding a monopoly on work in this field in the article) would sit in judgement on their fellow men to decide, before any work was done, whether or not it would be desirable.

An odd thing about research of any sort is that you never know in advance how the work will come out. If you know already, then you're not doing research. You are engaged in politics or a swindle of some sort. Any committee that meets to decide on the value of research before the work is completed is at a practical disadvantage. They can't possibly know what they're talking about. All of our decisions in advance of basic research are abitrary. Sometimes we luck out, sometimes we don't.

There's another practical problem involved in suppressing anything people really want to do. (If you haven't heard about the excitement that went with national prohibition, you may find a number of interesting books on the subject in the library.) It isn't terribly difficult to find folks who are brewing whiskey, for example, because the cooking stuff smells so good for miles away and because the necessary physical plant is rather bulky. So are the supplies. The real key to the moonshiner's success is bribery, not secrecy.

In case you hadn't noticed, transistors are a lot smaller than bags of corn and sugar. A whole lot of transistors, thousands of them in thoughtful combinations, can be arrayed on integrated circuits. They operate quietly and they all look alike, no matter what they are doing. The researcher into artificial intelligence needn't build a big fire under a kettle. He is therefore hard to catch.

If the neighbors were to develop robotics and you didn't, you might be at some disadvantage in figuring out how to handle the neighborhood robots. They'd be on your doorstep, demanding your attention. Would they go off and play elsewhere if you gave them cookies? Would yours, if you had any? By the end of 1975 a revolution had occurred that made robots inevitable, for good or ill. We'll learn which after the fact.

Robots *will* be on your doorstep, no matter what the elitists think.

In 1975 computer technology slipped forever out of the control of the professional establishment and into the hands of amateurs who were eagerly waiting to get computers of their own. A lot of professionals pooh-poohed the ameteur computer, pointing to the limitations of these small systems built around newly available "microprocessors", but the amateurs who had been thirsting for a chance to fool around with real computers of their own didn't laugh, they bought . . . and bought.

There are no experts yet in the field of amateur computers, but those who aspire to the experts in years to come guess that two to three hundred thousand computer hobbyists will be active in clubs by 1980.

Now then, who will prevent these enthusiasts from innovating as they please? Who will explain to them just what it is they may not question, where they may not search, what they may not try? Who, in fact, will be able to figure out *what* they *are* doing?

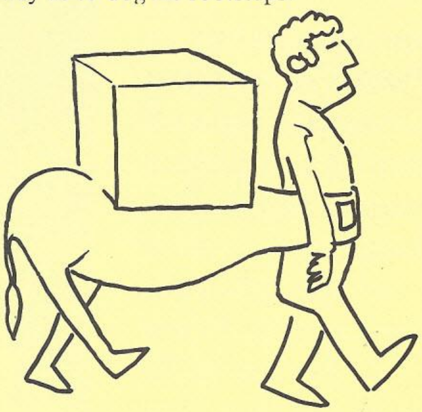
The amateurs can take far greater risks than professionals. A commercial enterprise that pours all of its resources into a product that fails in the market may fail right along with the product. What does an amateur have to lose? Time? Reputation? So what? Even a fuzzy-wuzzy intellectual, safe with tenure in a university or a government lab is subject to unpleasant reprisals from his management and colleagues if he inadvertantly does something that is not respectable in this tight little society.

No, the amateur is far better off. He can do what he likes. Some amateurs like robots and are working on them. Want to create a dedicated undergound movement in nothing flat? Try to stop the amateurs from working on robots. Robots are inevitable. 1975 was the turning point.

FROM CHAPTER 2

A discussion of Animal Intelligence and Ways of Employing it.

. . . How about designing a beast of burden that uses a man's own reflexes and low-level skills to carry a load? Let this pack animal, this ass, be secured to the walking man himself in such a way as to dog his footsteps.

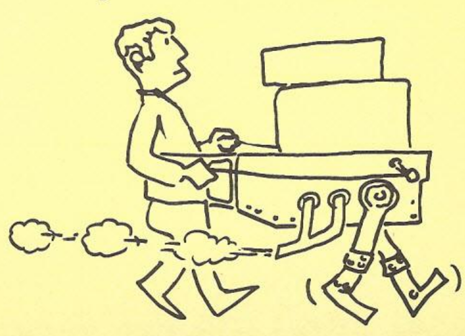


The appendage, sort of a prosthetic ass, would have its own set of walking legs over which the greater part of the pack burden would be placed so that the man himself carries very little. The legs of the prosthetic ass would march right along with the man, engine powered, moving at his pace and with his own gait. The legs would be controlled by the movement of the man's legs, but not necessarily in step with him. Probably their timing would be adjustable, controlled either automatically, by sensors, or by the man, so that the whole centaur-like structure would not oscillate rhythmically and upset itself.

Climbing uneven paths and hills, the man might wish to take over more thoughtful control of the prosthetic ass, guiding its pace and steps to maximum useful effect. The hind legs might even help shove him up the hill.

Indeed, there is no compelling reason to keep the ass walking behind all the time. It might be easier for the man to guide the thing in difficult terrain if it walked in front of him.

The possibilities are obvious.



Now you can buy an Altair™ 8800b or an Altair 680b computer right off the shelf. Altair plug-in boards, peripherals, software and manuals are also available. Check the list below for the MITS dealer in your area.

off the shelf.

ALTAIR COMPUTER CENTER
8105 SW Nimbus Ave.
BEAVERTON, OR 97005
(503)-644-2314

COMPUTER KITS
1044 University Ave.
BERKELEY, CA 94710
(415)-845-5300

THE COMPUTER STORE
820 Broadway
SANTA MONICA, CA 90401
(213)-451-0713

GATEWAY ELECTRONICS, INC.
OF COLORADO
2839 W. 44th Ave.
DENVER, CO 80211
(303)-458-5444

COMPUTER SHACK
3120 San Mateo N.E.
ALBUQUERQUE, NM 87110
(505)-883-8282; 883-8283

ALTAIR COMPUTER CENTER
4941 East 29th St.
TUCSON, AZ 85711
(602)-748-7363

ALTAIR COMPUTER CENTER
611 N. 27th St. Suite 9
LINCOLN, NB 68503
(402) 474-2800

ALTAIR COMPUTER CENTER
110 The Annex
5345 East Forty First St.
TULSA, OK 74135
(918)-664-4564

ALTAIR COMPUTER CENTER
5750 Bintliff Drive Suite 206
HOUSTON, TX 77036
(713)-780-8981

COMPUTERS-TO-GO
4503 West Broad St.
RICHMOND, VA 23230
(804)-355-5773

MICROSYSTEMS (Washington, D.C.)
6605A Backlick Rd.
SPRINGFIELD, VA 22150
(703)-569-1110

THE COMPUTER STORE
Suite 5
Municipal Parking Building
CHARLESTON, W. VA. 25301
(304)-345-1360

THE COMPUTER ROOM
3938 Beau D'Rue Drive
EAGAN, MN 55122
(612)-452-2567

THE COMPUTER STORE
OF ANN ARBOR
310 East Washington Street
ANN ARBOR, MI 48104
(313)-995-7616

THE COMPUTER STORE, INC.
(Hartford area)
63 South Main Street
WINDSOR LOCKS, CT 06096
(203)-627-0188

CHICAGO COMPUTER STORE
517 Talcott Rd.
PARK RIDGE, IL 60068
(312)-823-2388

GATEWAY ELECTRONICS, INC.
8123-25 Page Blvd.
ST. LOUIS, MO 63130
(314)-427-6116

ALTAIR COMPUTER CENTER
DAYTON, OHIO 45403
(513)-252-6785

COMPUTER STORES
OF CAROLINA, INC.
1808 E. Independence Blvd.
CHARLOTTE, N.C. 28205
(704)-334-0242

THE COMPUTER STORE
269 Osborne Road
ALBANY, NY 12211
(518)-459-6140

THE COMPUTER STORE
OF NEW YORK
55 West 39th Street
NEW YORK, NY 10018
(212)-221-1404

THE COMPUTER SYSTEMCENTER
3330 Piedmont Road
ATLANTA, GA 30305
(404)-231-1691

MARSH DATA SYSTEMS
5405 B Southern Comfort Blvd.
TAMPA, FL 33614
(813)-886-9890

mits

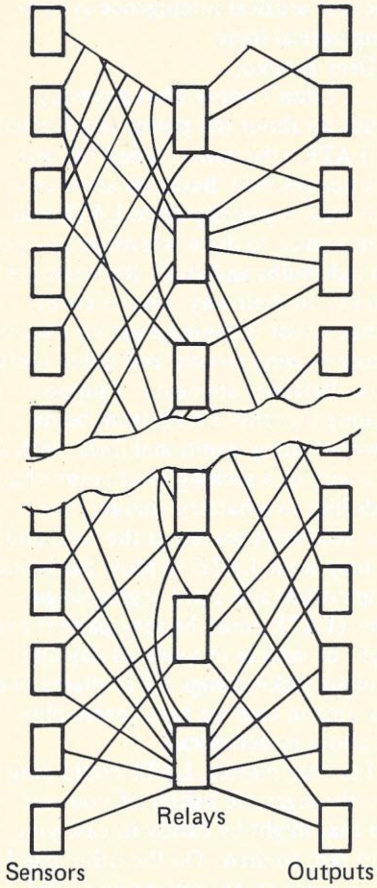
CIRCLE 36

FROM CHAPTER 3

A discussion of the Operation of Brains and Simulations of Them.

. . . One very useful computer simulation that we developed might be applied practically.

Fourteen sensors were arrayed at the top of this system, hooked up with fourteen outputs. The circuits all went



through eight neuron simulations in the center of the system, sharing them randomly. (This logical neuron simulation is discussed in detail and a program appears earlier in the chapter.)

Now, imagine that the sensors all detect what is happening in the real world. (Say that they are microphones) Each puts constant-amplitude, frequency-modulated pulses into its circuit.

Given a more or less steady level of random activity in the real world, the outputs down below all babble away randomly at the same level of activity. They may actually be connected to loudspeakers that make noise in response to action of the sensors.

Suppose now that something of special interest happens at one of the sensors, something "significant" from the point of view of the guy who set up this system. The pulses from that sensor grow more frequent. Not only does the speaker hooked into that sensor make more noise, but *all of the other outputs are blocked.*

The "important" thing preempts the system and calls attention to itself in no uncertain terms.

Could you do this with fourteen wholly separate circuits, tying them together only with a system that squelches the others when one circuit develops activity above a certain level? Sure you could.

The big factor here is the *reliability* of the neural net system. You can actually knock out *any one* of those eight neuron simulations in the middle, *randomly,* and the system would still work. Indeed, those eight gadgets in the center may be just simple relays, nothing at all fancy, and if somebody shoots one out with a rifle, the system will continue to work.

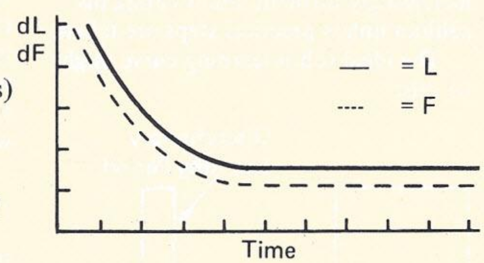
Reliability, combined with an ability to discriminate between things that are important and things that are not, based on experience, is a key feature of the human brain. That's what we use to deal with the changing world.

FROM CHAPTER 4

Some observations on robot learning and forgetting.

. . . It is a postulate, not a demonstrable fact, that in a human being, the rate of learning divided by the rate of forgetting is a constant . . . on the *average*, though the rates of learning and forgetting do vary throughout one's life. life.

The curve may look something like this:



d is rate
L is Learning
F is Forgetting
K is "a constant"

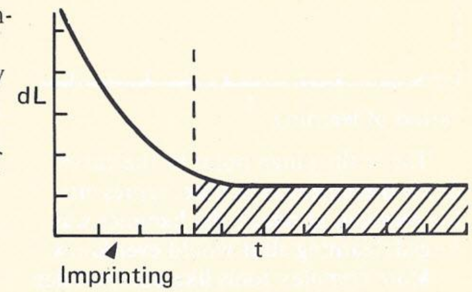
$$\frac{dL}{dF} = K$$

If L is larger than F, then K is greater than one.

Notice that a young person forgets things very quickly and needs a great deal of reinforcement to get ideas firmly fixed in his little headbone. As time passes, forgetting decreases, so that old people may retain a great deal of information without learning very much that's new. Old dogs remember old tricks.

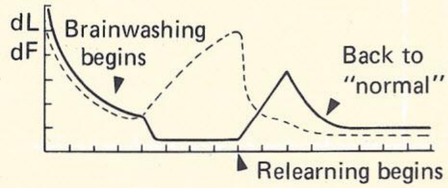
The trouble with slavery on a practical basis is that the learning curve in the human being never gets down to zero. You keep right on learning, all your life. We have never found a mechanism to prevent slaves from continuing to learn.

The learning rate is extremely high in the period of imprinting, when our basic notions are fimly developed.

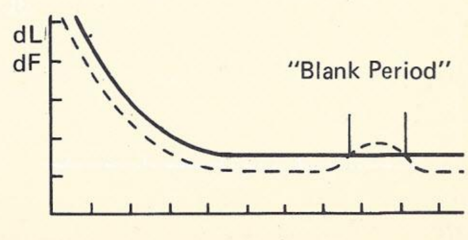


The shaded area is the zone of trouble with slaves.

This pattern can be altered artificially, with brainwashing techniques, possibly aided by enzymes that speed the destruction of existing cholinesterase/cholineacetylase, increasing the forgetting rate dramatically.



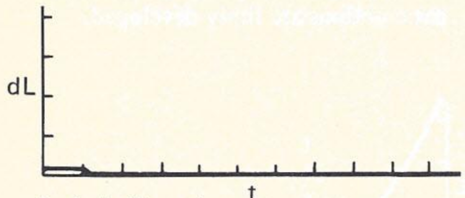
Minor aberrations of this sort undoubtedly occur naturally, anyway, as the learning and forgetting rates change slightly. For various reasons, the forgetting rate may develop a little bump that carries it above the learning rate curve. In that time, perhaps during a famine, forgetting is greater than learning.



This may appear as a 'blank period' in memory later.

We have not traditionally equipped artificial slaves with "potential" capabilities, weak circuits that may be reinforced or not. Usually, we build things to do *particular* tasks well and if these implements are applied to other tasks, they usually do them poorly. (If they do other things well, it's a happy surprise.)

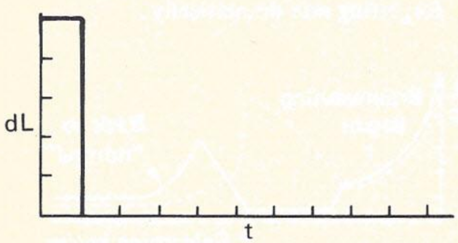
A hammer, for example, is made to transfer energy in sharp shocks to a selected small area in some other stucture.(Some of us have been known, in desperation, to use hammer claws as screwdrivers, but that application of hammers is incidental and unsatisfactory.)



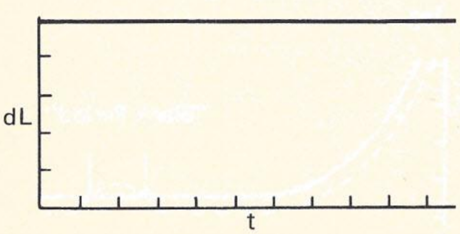
Period of learning

The trifling high point in the curve, down there near zero time, represents the moment at which the hammer was forged, learning all it would ever know.

More complex tools like automobiles and washing machines are able to carry out a series of complex operations, once they are set in motion. We take some time and care to build these features into complex systems. The learning curve might be:



For the washing machine, this represents creeping failure. It isn't intelligent. It's too dumb, has no flexibility, can't do anything appropriate under unexpected circumstances. On the other hand, if the curve were this:



the failure would be catastrophic, because the machine would be too flexible, responding to change so willingly that it would be unable to act in a constantly changing world.

And what about robots?

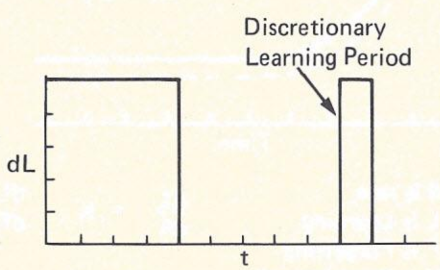
If we hope to use them and not be used *by* them we must prevent them from continuing to learn after some optimum point.

After the robot babysitter has learned to carry out certain instructions with respect to the kid ("Don't let him run in the street or fall from high places." "Make him eat his pablum."), you don't want instructions that require the babysitter to *enforce his will* on a child who may be resisting eating his pablum; you want the robot to *lay off.* You don't want the robot to choose the stories to read to the kid, imprinting the infant with robotish ideas that make him think like a robot in later life.

You don't want the robot on your factory assembly line to volunteer little helpful actions for you. If the robot has seen jars coming off the end of the line with caps on them, he may decide to help you out by putting caps on the jars he is merely supposed to be washing at his place on the line. This will make it more difficult for robots down the line to insert the pickles they are supposed to be packing in the clean jars. No end of confusion will result if the robot learns more than he absolutely needs to know.

On the other hand, after the child has grown up or the pickling season is over, you'd like to be able to modify the robot's knowledge so he can do more useful work, not necessarily forgetting what he has learned in the past (though you may wish to keep him from calling the *second* Mrs. Smith by the name of the *first* Mrs. Smith), but expanding his knowledge in certain ways. The fifteen-year-old who grows out of the original infant will become increasingly difficult about eating his pablum unless practical steps are taken.

The ideal robot learning curve might be this:



Whatever you do, don't leave the learning switch on. What you don't need is a robot stronger and smarter than you are. These little graphs have not included the forgetting curves or a number of other things that might be appropriate for discussion, but you get the idea anyway.

FROM CHAPTER 5

"The Conrad Correspondence", a series of letters specifying the design of a practical artifical intelligence system using optical logic.

Dear Brocko,

. . . I don't know what anybody would do about the power requirements for LATE, (the name coined for this robotics system). Batteries are nice in their way, especially since LATE will require juice to drive television systems and lightbulbs and stuff. Batteries are also bad in their way, since a cheery, mobile robot is going to use up a lot of energy moving around and doing useful work. Batteries are heavy. I suppose nothing prevents LATE from being powered along traditional lines, with a long wire or a seeking mechanism that leads him to a battery charger, but that does limit his use in the field and we may want LATE to plow fields and chop cotton and weed vegetable gardens. (LATE could be very patient and might be willing to work all day and all night picking bugs off of plants one at a time in case we have severe phobia about insecticides.)

For that matter, LATE could plug into the cigarette lighter of your car and that might be handy in case you want him to drive. On the other hand, if you want an exceptionally stupid, I mean limited, version of LATE, a stripped model as it were, to do nothing but drive, you could probably order him as factory equipment on the car and have him installed like air conditioning. It seems likely that you'd want a factory school to train him, too. Otherwise you may teach him to be as bad a driver as you are yourself and that would never do.

Oh yes, power, we don't always need a lot of power.

One can imagine situations in which speed is not the primary concern. We may want things to be done in real time, but real time does not always unfurl at electronic speeds.

Suppose, for example, that you are an Indian tribe with a reservation on the plains, up against the mountains. You are interested in maintaining a big old pasture on the plain near the rail-

M.A.C.C. COMPUTER 1977 CONVENTION

The Midwest Affiliation of Computer Clubs invites you to attend the Second Annual, Midwest Regional Computer Convention and Exposition....

JUNE 10, 11, 12

COMPUTERFEST '77

BOND COURT HOTEL *777 St. Clair, Cleveland, Ohio*

- > Manufacturers' Reps & Exhibits
- > Roofed Flea Market
- > Seminars & Tech Sessions
- > Club Congress
- > *And Prizes, Games, Media Duping, and more....*

Still only* 2.00 *per ticket

The M.A.C.C. Convention is Where It's At in the Midwestern region. See you there! And if you plan to attend NCC, why not fly with us? Ask for information about the charter flight to Dallas. Drop a line to:

MIDWEST AFFILIATION OF COMPUTER CLUBS

P.O. BOX 83, CLEVELAND, OHIO, 44141

road so that you can fatten up your cattle and sheep near the transport site.

The quality of the grass and other growing stuff in this pasture is determined by the way things are up in the hills . . . how much snowfall has stacked up during the winter, how many beaver dams have been built lately and how many have been broken up, how much light gets to the brush under the trees, and all those other natural things that make up a local ecosystem.

Presumably, given patience and an array of small techniques, each of which has a minor effect on the local situations, a body could very strongly influence local conditions without doing anything dramatic or violent.

Obviously, you can dig irrigation ditches and pump water here and there, but it isn't always socially acceptable and it isn't always economically or physically feasible. If you're an Indian, living in harmony with nature, you will be torn painfully between a desire for consistently fat cows and a desire to hang in there with Manitou and play the game properly. Digging things up with bulldozers may be bad news, but encouraging or discouraging the growth of light or dark leafed plants selectively may be OK. Raising or lowering the water level of a bunch of small ponds by a few inches may be all right, making noises to attract or frighten away certain animals may be all right, raising or lowering the temperature a degree or two on the average may be acceptable.

To get to the point . . . LATE could live quietly in a brackish pond, soaking up energy from the sun, thinking quietly about how things are and making minor adjustments to maintain or change them.

Given that we might construct a device that simulates thinking processes using lenses and bits of wire and flakes of crystal and light sensitive material and other such trash, why might we not also use living material . . . bacteria, algae, and so on, in the device?

These living things already have a lot going for them. They manage to survive over the millenia by doing their own thing, finding food, reproducing. From the viewpoint of a machine builder, it would be nice to work with components that organize themselves, protect themselves, adapt to changing conditions, and keep right on doing their work, no matter what.

Biological systems don't always "act" quickly. It takes a while for a colony of bugs to respond to a change. For example, if you blow noxious fumes like tobacco smoke over a colony of luminescent bacteria like the kind that grow on rotting fish, the bugs faint and quit glowing. Then they recover and over a few seconds come back up to strength and glow brightly again.

If we built an optical logic system that used these bugs for a light source that is also a sensor, the system could work just fine, but not very fast. We probably wouldn't want to use such a logic system in driving a car, but if we have all week to figure out what's going on and make some response, the system may be perfectly adequate for our purposes.

Maybe the flickering of the aurora has addled my brain up here, but when Mailer talks about all this it makes some kind of sense to me. I can see LATE as a box of stuff submerged in a peaceful pond amongst the muskrats, connected by wires to many other boxes here and there in the hills. Associated with the hardware are bags of bacteria, warmed and nourished by the sun and streams.

At times, Indians gather up buffalo chips or whatever else there is around the hills and plains and bring it to compost heaps near the pond where LATE works. LATE can draw chemicals from this to feed the bugs or can withhold this nourishment to limit their numbers.

Sensing water level, temperature, wind, strength of the sun, density of undergrowth, thickness of tree rings, and all those sorts of things, LATE could raise or lower sluicegates a trifle at hundreds of locations, could release enzymes into the water that stimulate or suppress the activity of the beaver, could encourage or suppress the growth of narrow leafed grasses or broad leafed skunk cabbage, could vary the absorption or reflection of infrared radiation from the sun.

Quietly working away through the passing seasons, LATE could tweak up the whole area so that the pasture is properly watered and fertilized for them there cows and sheep.

I don't know whether this is appealing or repellent, but it certainly does seem possible. On the whole, it seems to work more *with* nature to encourage life and harmony than against it. You Indian types might find it just the thing.

Not having a Xerox machine here in the tundra, let alone enough power to run it, I have torn some pages out of a recent journal. The pages contain an article about bioengineering which will suggest to you why this seems practical.

There go the lights again.

Personal Computing comes to Chicago

- The first Midwest Personal Computing Show – Thursday, Friday, Saturday – October 27, 28, 29
- Exhibits of personal computers for the home, small business and school
- Also – numerous workshops, seminars and special events

For more information watch for the July/August issue of **PERSONAL COMPUTING** *magazine or write to Austin Cragg*

CONFERENCE & EXPOSITION MANAGEMENT COMPANY, INC.

Post Office Box 844 • Greenwich, CT 06830 • Tel. (203) 661-6101

FUTURE COMPUTING GAMES

by Rick Loomis

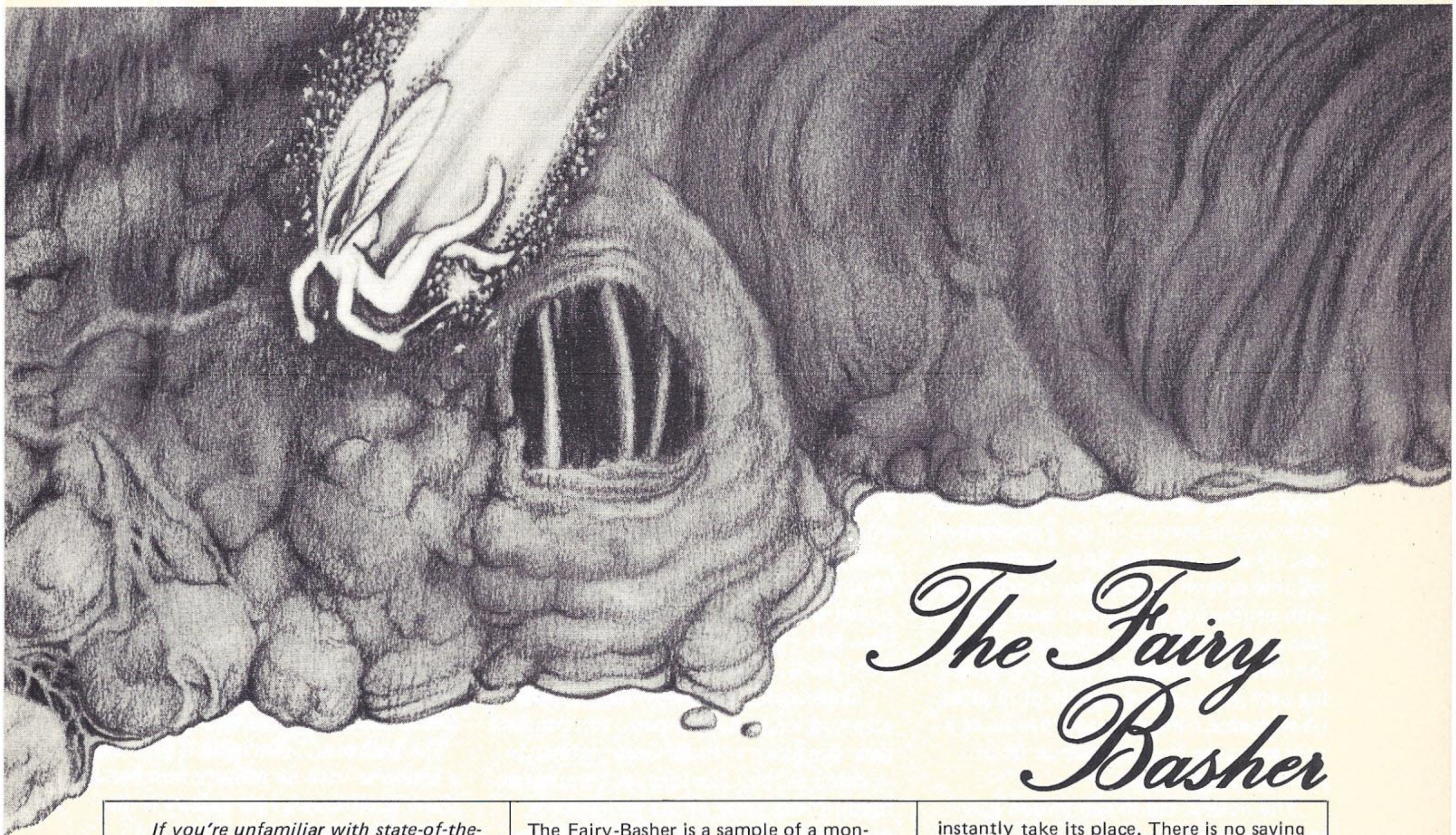
Rick Loomis is unique in earning a living as a fulltime "game moderator." He is proprietor of the redoubtable Flying Buffalo, Inc. in Scottsdale, a firm that uses computers to keep track of complex games with active players spread hundreds of miles apart. PERSONAL COMPUTING asked the Buffalo to gaze into the future of computer gaming and report on what he sees. Here is the report from the desert.

First of all, remember that my primary field is conflict-gaming (the conflict being economic, political, military or some combination thereof), so I will be commenting chiefly on this kind of gaming.

I see two major kinds of computer games in this field. As people get more interested in sophisticated games and as computers get less expensive, eventually the two will meet to create my two dreams. The first is the ultimate two-player game. At least 100,000 people in this country (by most estimates) are interested in military conflict-simulations: in other words, wargames. As most of these games are billed as "realistic simulations of historical events", there is a real problem of playability versus realism. If your wargame were totally realistic, you would have to keep track of such logistical problems as how much food, bandages, ammunition, shoes and blankets your troops have. But if you want to be able to play the game, you would like to ignore all these problems and perhaps add some kind of supply rule that pretends to simulate logistics. (Example, you must have a "supply unit" within 5 squares of each of your combat counters or they will fight at half strength.) The more complicated rules you add to your game, the more realistic it is and the less playable!

Another major problem is being able to see all the units on the board. Most of the games have all the pieces on the board at all times. When it is your turn, you can sit and count squares to see exactly where any enemy unit can be at the end of the turn plus exactly how strong each enemy unit is. Some games try to make up for this by introducing special rules such as simultaneous movement, counters that are upside down until they enter combat, counters with variable strengths, dummy counters that the other player can move to confuse you and so on. Again, the more such rules you add, the more difficult the game. Simultaneous movement works great, for instance, as long as there aren't more than 20 or so pieces per side. The big games with 2000 pieces are simply impossible to play simultaneously. The ideal solution is to have three games, with a third player acting as the referee. You move your pieces on your board and your enemy moves his pieces on his board. The referee keeps track of both sides on his board and tells you when you sight an enemy unit. But this requires considerable time, extra copies of the game and finding someone willing to act as the referee.

Hidden movement is the best place for the application of the personal computing. I hope to see soon a micro sys-



The Fairy Basher

If you're unfamiliar with state-of-the-art gaming, you may be surprised by the immense complexity of current games and the richness of color with which the background detail is woven. Computers are able to keep track of the background as well as player activity, displaying or printing out material that strongly enhances the escapist reality of the game. Consider this detail from Dungeons and Dragons:

The Fairy-Basher is a sample of a monster which might be in a dungeon. He is designed to catch unwary players who are playing the part of a fairy. The Fairy-Basher hangs around entrances of dungeons and builds sophisticated air blowers, which knock unsuspecting fairies out of the air and onto the floor. When a Fairy-Basher sees a fairy which has fallen onto the floor, he comes running out of his lair and stomps on the fallen creature with his big feet, then eats the remains. Usually found in groups of 1000, Fairy-Bashers are very fast. If you kill one, two more instantly take its place. There is no saving throw (once you fly into that air-blower, you've had it!) If you try to carry a fairy into the dungeon in your pocket or knapsack, the Fairy-Bashers smell it and run out and start gnawing on your feet in an attempt to get at the little critter. The amount of damage done to your feet and legs depends on how much the Dungeonmaster hates fairies, and how long you stand around trying to fight instead of running! (Fairy-Bashers just hang around the entrance, and do not follow you into the dungeon).

tem available where each player has a keyboard and a CRT screen. Any section of the map can be displayed on the screen, at a command from the player. Units are displayed in their locations, with exact locations of your own units listed by coordinates at the bottom of the screen. The computer gives you condensed reports of the action along with intelligence reports, spotting, casualty lists and anything else you need to know and ask for. It asks you for orders for each unit and the detail required depending on the game. (Are you playing a division commander or a company commander?) You are told about only the enemy units which are in sight of your own units, thus allowing the players to make surprise attacks, feints and other military maneuvers not often allowed in regular wargames. Instead of being able to count up the exact enemy strength, you will be given only estimates depending upon how much contact you have had so far. The computer could even time your moves so that the faster you move, the better your results (assuming you made the correct move, of course.).

There must be 800 or 1000 different wargames on the market today, representing practically every major battle in the history of man. Each one could be programmed separately and sold on cassettes to the owners of the game-system described above. The first company to sell this system will be able to get away with producing one or two new programs a year. When competition enters the market, companies will be producing new games every couple of months. (A wargame company now produces a new wargame in each issue of its bimonthly magazine).

The newest fad in simulation gaming these days is a fantasy game called DUNGEONS & DRAGONS, and several spinoffs of it. This game is played with pencil and paper. One player (called the "dungeonmaster") creates a dungeon. He usually draws it on graph paper, placing corridors, rooms, doorways, secret doors and passages, and so on in any pattern he chooses (or in no pattern at all). Then he scatters traps, monsters, tricks, treasures and prizes all around his dungeon (usually in groups)

trying to kill the monsters and find the treasures. I can imagine what the above computer system could do with this game, especially with the color tv screens now available for most systems. I'd like the system to display on the screen what you as the player would see if you were really in the dungeon. Imagine: the screen shows a dank, dark corridor with green slime on the walls and flickering torches lighting the way. You instruct your character to walk down the corridor and you watch as the walls slowly move by you. Suddenly, you come around a bend and are face-to-face with an Ogre carrying a large club. What do you do? Fight, run in terror or just stand there in shock while the Ogre hits you over the head with his club! Oops. Go back to start!

Once again, when the basic system is introduced, the company can produce new dungeons with different monsters as often as they can manage to program them. If the price of the programs is reasonable ($5 to $15 each), the company will have a steady income (not unlike a book-of-the-month club.) And as crazy as the current players are with the paper and pencil version of the game, they will just go wild over a system like this. Already, individuals are combining the game with personal computers for several aspects of the game. But it will take a sustained effort by someone with money and vision to come up with the ultimate version of this game.

I did say I see two kinds of computer games. The second is closer to reality, and one which my company is primarily working on. This is the giant, multi-player, super complicated game where the computer is in one central location and the various players mail, phone or telegraph in their moves. I can see a huge wargame with a map of the world representing all countries. Small countries are played by a single player, while the large countries are played by teams or groups of players (including players representing the "loyal opposition", subversives and so on.) Some players could run crime syndicates, others could be religious leaders. You would have at least 500 or 600 people involved in one game. No one player would have the entire rule book – each one gets only the set of rules which directly involve his particular character. This would not be an ordinary game in the sense that someone eventually wins. Very possibly the game could go on forever. When a player dies or gets tired of the game, his position would be auctioned off or presented to the next person on the waiting list. It might even be something you would list in your will. ("And to my nephew, Fred, I leave my position as ruler of Sweden in "The World" game.) This game would have wars, economic, politics, shady deals and natural disasters. It would have room for people who want to build empires and people who just like to make trouble for others. (Position paper: *You are the leader of the railway workers union in Latveria. You are also a secret intelligence agent for the neighboring country of Borondigas. Your mission is to cause a crippling strike just before the planned invasion.)* The game would require quite an extensive computer system, of course! This is the game I want to play; no one else seems to be offering it at the moment. Eventually we will get something like this going. Of course, we can't stop there. The next game will be called "The Universe". (Imagine if you will, a Universe with an infinite number of stars. Each time a new galaxy is discovered, we buy a new disk . . .)

By this time, you're probably asking yourself, "why would anyone want to play these crazy games anyway?" Well why not? Here's your chance to be president, dictator, millionaire, tycoon, starfleet commander or emperor. So you lose. So what? Just start another game. Start three more. Sooner or later you will win (and how sweet it is!). I've always wanted to conquer the world. What's your dream?

In the January/February issue of Personal Computing, Dick Heiser looked into his crystal ball to find out what he would be selling in his Computer Store two years hence. Here's a point-by-point response from another leader in the field.

Dear Dick

. . . I thought I'd give you a little feedback on your article, which struck me as an excellent prediction! (i.e. — we seem to have some similar viewpoints and perhaps a comparison of viewpoints would be interesting to us both).

I took the liberty of writing, not to point out "where you blew it," but as a reaction to our apparent agreement. Almost in every case, the only difference is in amount or timing.

I also realize your article was written some months ago and that events might be happening a little quicker than you realized at the time. Big deal! Still a very impressive article!

Richard C. Bemis, President
The Digital Group
Denver, Colorado

Article Point	Comment
CPU	
8-bit Z80 type	Agree for time being.
16-bit 9900	Also agree for number crunching — very small market.
Memory — 16K average	Low. Standard average probably 32K shortly.
1st 1K ROM	Agree.
Short, wide 24-bit memory bus	Agree.
Low-order 16 bit	Agree.
Long, narrow I/O bus	Agree.
Real-time clock	Agree, several simultaneously with special precisions.
Price $700	High.
Mass Storage	
Digital Cassette	Of course, but mostly in addition to disk.
DECtape	Agree.
Data re-written in place	Maybe. Some problems here.
Density — 800 bpi	Low.
Parity/CRC	Of course.
Fast search — 20 secs.	Agree.
Read after write	Yes.
250.00	Probably.
Disks -dual @2,000	Way too high, even for 2 Mbytes. There is an awful lot of action here now. And lots of price movement.
Audio tape.	Starter systems only.
Keyboard	
Reliability a must	Yes, yes, yes! Solid state types only need apply.
Upper/lower case	Text entry will force this as you state.
Keyboard layout	Standard typewriter *far* more popular.
Large format func.	Don't know about this one.
Price — $150	Exactly correct for today — probably $50 higher before 1978.
Video Display	
16 x 64	Agree — minimum.
RF hookup	Agree, but obviously inadequate for dense displays.
Vector graphics	Yes.
Integrated video	Yes.
$150	High, one chip soon.
Video Monitor	
High-Resolution 9" for $200	Agree. Ah, the power of mass markets!
15" for $325	Agree. but perhaps a little high.
Printer	
Badly needed	Absolutely agree that printers are currently a disaster! However, if we can believe what we've seen in prototype form coming to production on schedule, then some fairly good answers will soon be available that make sense at quite reasonable dollars. Again, a great deal of low-level activity going on here. Printer manufacturers are not dummies — dam should have broken about summertime. (We sure hope so!)
Selectric	Your evaluation coincides exactly with our stillborn product for the same reasons.
Matrix	The only economical hope at this time.
Missing features	
Interrupts needed?	Absolutely not. Waste of time and effort.
Power, fail & detect not needed	Agree. Magnetic storage recovery perfect for most apps.
Front Panel not needed	Couldn't agree more!
Multiprogramming not needed	Again yes! Economics dictate purchasing another machine. Why strain to share a $2000 resource?
Storage protect not needed	Agree. Waste of space.
Options	
Mass storage	All applicable, of course. Growth path should allow immense headroom for development.
Other	Yes.
Improvements	
Color graphics	Agree software too complicated. (Another stillborn here.)
Parity checking	Late in development cycle — agree it's needed.
Budget	
$2450	Probably too high, or not enough capability for $.
Overall comments	
Quality	Agree.
Assembled domination	Agree.
Service contracts	Some.
Dealers sales to dominate	Yes.
Dealer variations	Yes, all catagories and others as well.

Digital Illogic Made Logical
or
does that stuff really make sense?

by Steve Pollini

No doubt at one time or another you've attempted to plow through a article that described a device (like a personal computer) that used digital logic circuits. To most of us such articles generally appear to be composed of a random assortment of digital illogic and unfriendly jargon. The mystery of these technical papers can be cracked, even by the casual reader armed with a little information and relieved of his awe. Digital logic can even become entertaining.

To begin, we'll discuss the symbols used in describing digital logic. They're straightforward, since they deal only with ones and zeros; and not any complicated numbers like two, three, four, etc...Entire computer systems are built upon circuits which manipulate ones and zeros.

What's the simplest, most elementary operation we can perform upon the number one (1)? Why change it to a zero, of course! That's called *inverting* the number. Inverting a one changes it to a zero; inverting a zero changes it to a one. This is generally referred to as the NOT Function. The symbol which describes this inversion looks like Fig 1 and the truth table shows us the possible combinations.

A	B
0	1
1	0

Fig 1

As you can see, when a one is put in, out pops a zero; when a zero is put in, out pops a one. You're now on the road to becoming a digital logic expert.

Another function used in digital logic circuits is the AND operation. The symbol used is in Fig 2.

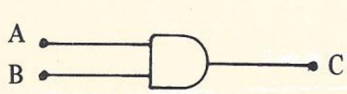


Fig 2

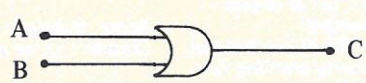
The AND gate has two or more inputs (A and B) and one output (C). In order to have a one at C, there must be a one at both A AND B, otherwise C will be a zero. The truth table below describes all of the input and output combinations possible.

INPUTS		OUTPUT
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table For AND Function

Notice here that when there is a one on only A or B that there is no one at C. It is only when *both* A AND B are ones that a one appears at C. This function can also be represented notationally as: $A \bullet B = C$. The dot between A and B indicates the function AND.

Here's another, called an OR gate. The output of the OR gate will have a one when either A OR B is a one; the truth table shows all of the possible combinations at a glance in Fig 3.



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

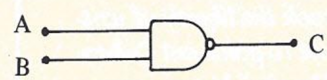
Fig 3

Here it becomes obvious (only to those who have pondered this for years, of course) that a one will appear at C when either A OR B OR BOTH are a one. Notationally this is represented as: $A + B = C$

The plus sign means OR *not* AND. Remember that we use a " • " to designate the logical operation AND.

On to bigger and more profound revelations . . . .

Now we're going to combine what we've learned about the INVERTER and the AND gate. The symbol below represents the NAND (NOT-AND) gate.



The little circle at the end of the AND gate means to invert the output of the AND gate which makes the whole thing a NAND gate. Take a close look at the truth table to see how this happens.

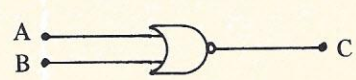
A	B	C
0	0	1
0	1	1
1	0	1
1	1	0

In the first condition, when two zeros are ANDed together to produce a zero, they are then inverted to produce a one. Remember also that when a one and a zero are ANDed together they produce a zero. Inverting this zero gives us a one at C, the output. The notational representation of this is very interesting.

$$\overline{A \bullet B} = C$$

Here the " • " means to AND A and B and the bar over them means to invert the result. Thus C equals NOT (A AND B).

The very same principle can be applied to the OR gate to produce what we call a NOR gate.



In this gate A and B are first ORed and then INVERTED to produce the output C.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

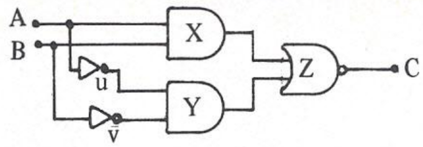
The truth table shows that when two zeros are ORed they produce a zero which is then inverted to give a one at the output. This is the only case in which a one will appear at the output, because all of the other conditions produce a one which is inverted to a zero at the output. This function is notationally represented as follows:

$$\overline{A + B} = C$$

A is ORed with B, then the NOT function inverts the result to produce C.

Let's now investigate how we can hook up these basic logic functions to produce circuits that perform even fancier operations.

To produce a circuit known as the EXCLUSIVE OR gate we need inverters, AND gates, and a NOR gate. An EXCLUSIVE OR gate produces an output only when A OR B is a one but not when both of them are ones.



A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

To understand what is really going on it would be good to trace through the entire circuit with A and B in each of the conditions shown in the truth table.

When A and B are both zero, AND gate X produces a zero. Inverters U and V invert the zeros to ones which, when ANDed by Y, produce a one. The zero from X and the one from Y are ORed to produce a one and then inverted to produce a zero by NOR gate Z. Thus the output is a zero when both inputs are zero.

When A is a one and B is a zero, AND gate X will produce a zero. Inverters U and V will feed a zero and a one to AND gate Y respectively. Y will produce a zero. The zeros from Y and X will be ORed to give a zero and inverted to give a one by NOR gate Z. The output at C is then a one when one input A is a one and the other input B is a zero.

Now it's your turn to do figuring. Try to go through this circuitry when A is a zero and B is a one, and when both inputs A and B are ones to verify what the truth table shows. The notational representation for the EXCLUSIVE OR gate is below.

$$A \oplus B = C$$

You can see that the standard OR symbol is used with a circle around it which makes it an EXCLUSIVE OR.

The circuits and symbols that you have just learned are used every day by engineers and programmers; if you have worked through these little logical exercises, you have a good appreciation of the processes by which complex computer systems are developed from a handful of simple logical operations.

You needn't be intimidated by the technical talk and symbolism. With persistence, you can not only figure out what articles mean, but explain your own insights in the same terminology, giving as good as you get. You may go on from this modest introduction to a pleasant study of the general field.

PERSONAL COMPUTING'S LOS ANGELES SHOW

By Anthony Abowd

An hour before show time, hundreds of people jammed the registration area for PERSONAL COMPUTING'S first Western Computing Show, March 19, in Los Angeles. When the doors finally opened, they flooded the display areas and devoured literature. In two days over 3,000 personal computing fans saw a large selection of systems, peripherals and program packages. They attended seminars in software and hardware, played with the gadgets, listened to electronic music and generally enjoyed themselves.

and now, the megabyte machine

A major announcement at the show came from IMSAI, the San Leandro, CA computer manufacturer. The firm announced that the existing popular personal computer chassis, known as the Altair or S-100 bus, can now handle about one million directly accessible memory locations, called a "megabyte" of memory. This huge advance in memory capacity is possible using IMSAI's new bigger capacity memory boards.

"This announcement will blow everybody's mind," says Duane Hentrich of IMSAI. "We now have a 65K RAM board. This is a single board, the same size as previous memory boards which used to handle, at most, 16K." RAM, short for random access memory, is the fastest, most accessible computer storage. IMSAI engineers figure that before this new development, a megabyte of RAM needed a computer chassis about 16 feet long.

The 65K board sells for $3899 assembled, less for the kit form and smaller amount of memory. This price is probably out of reach for most personal computing fans, but somebody thinks it's reasonable. IMSAI sold 50 of the new boards in the first three hours after they were announced. The new boards also need a special memory management board which is $399, assembled. After the show several other companies announced similar RAM boards to compete with IMSAI.

seminars flooded

Many of the visitors to the first Western Personal Computing Show did not know enough about personal computing to realize the significance of the megabyte machine. This did not, however, stop them from flocking to the various seminars offered in conjunction with the show. Gene Murrow, president of Computer Power and Light (COMPAL), a Los Angeles computer retail store, conducted free introductory computer tutorials for enthusiastic, overflowing crowds every hour on the display floor. Tom Munnecke of Metasystems, a computer consulting service, delivered his "Personal Genie" free introductory seminars to large crowds. Patricia Wood explained how novices could build a computer from a kit. Rodney Zaks, who gives professional, classroom-type personal computing courses for Sybex, Inc., had 300 registrations for his $20 courses.

"We think PERSONAL COMPUTING is on track in bringing this kind of show to people who obviously have a hunger for it," says Gene Murrow. He sees the hobby computer market moving away from emphasis on technical aspects and kits toward the less sophisticated user.

"I'm a technical person and I think it's fine," says Murrow. "Some people's hobby is flying airplanes and other people's hobby is building computers. But another hobby, I think, is programming computers. I think that's a poten-

tially larger hobby because it's closer to what people associate with a computer, namely, a tool that can do things.

"We call this the second generation hobbyists. They don't want to build a computer and use it They want a tool. They almost don't care what's inside the box," says Murrow.

less switches, more programs

Almost every exhibitor at the show echoed Murrow's ideas about the direction of the personal computing market. Many companies introduced computers without the traditional array of flipper switches and lights on the cover. These switchless computers, called turnkey systems, are intended for the less sophisticated user. Internal programming takes over when the outside switches are left out.

One such new system is Cromemco's Z-2. "This is the first time the Z-2 has ever been shown," says Alice Ahlgren of Cromemco. "It's the only rack mount computer that I know of. Just stick it into a rack and forget about it. You get a blank front panel so that people, particularly kids, cannot get in there and flip the switches to destroy what's inside." Cromemco is interested, among other things, in producing a more durable computer for use in classrooms and industry. The Z-2 is $995, assembled.

Several other manufacturers are procuding switchless boxes. The Digital Group, out of Denver, Colorado, has built an entire system around its turnkey computer. They introduced a new printer at the show that prints 120 characters per second and 92 characters per line. The Digital Group is one successful personal computing manufacturer not using the S-100 bus. "By midsummer, you should see a complete line of Digital Group stuff to the point that you won't be too concerned about being locked into our bus structure because we'll be giving you everything you need for the best deal," says Steve Ingoglia of Digital Group. The Digital Group systems start at $1295 assembled.

PolyMorphic Systems, of Santa Barbara, CA, is one firm marketing a complete system for novices. They showed their top-of-the-line System 16, and S-100 compatible computer with 16K of memory, a keyboard, video monitor, cassette recorder and an expanded BASIC programming language for $1995. "This is fully assembled and tested. In ten minutes you should be ready to start programming. You are ready to go when it's delivered," says Cindy Feeny of Polymorphic Systems.

Vector Graphic, Inc., from Westlake Village, CA, introduced its Vector 1 computer at the show. "Maybe people are really interested in having a computer that doesn't have all the switches," says Lore Harp, president of Vector Graphic. "Maybe it's a little beyond the hobbyist who likes to fiddle with all that and likes to see everything work. People are interested in just plugging something in, hitting a button and going. It's like plugging in your television set or stereo. You select a station and there it is." The Vector 1 is $849, assembled, with the unusual feature of a cabinet available in two colors, green or orange.

field day for hobbyists

The show had its share of technical hobbyists. "Here at the show, all these guys have knowledge of electronics. They know either hardware or software and they've got friends who know the other side.
They can take an unstable product and live with it," says Herb Waite, marketing director of PerSci, a disk drive manufacturer at the show with the firm's new "intelligent" disk drive. "But for the home market, you're not going to be able to do that."

Mits, of Albuquerque, NM, was chiefly demonstrating small business applications for its 8800B Altair, but also showed off the smaller Altair 680.

"The Altair 680 went over pretty well because we are coming out with more peripherals to support it. It started out as a single user computer, for burglar alarms and so forth. We built it into a very powerful computer and we've put some things around to make it usable," says Chuck Olsen of Mits.

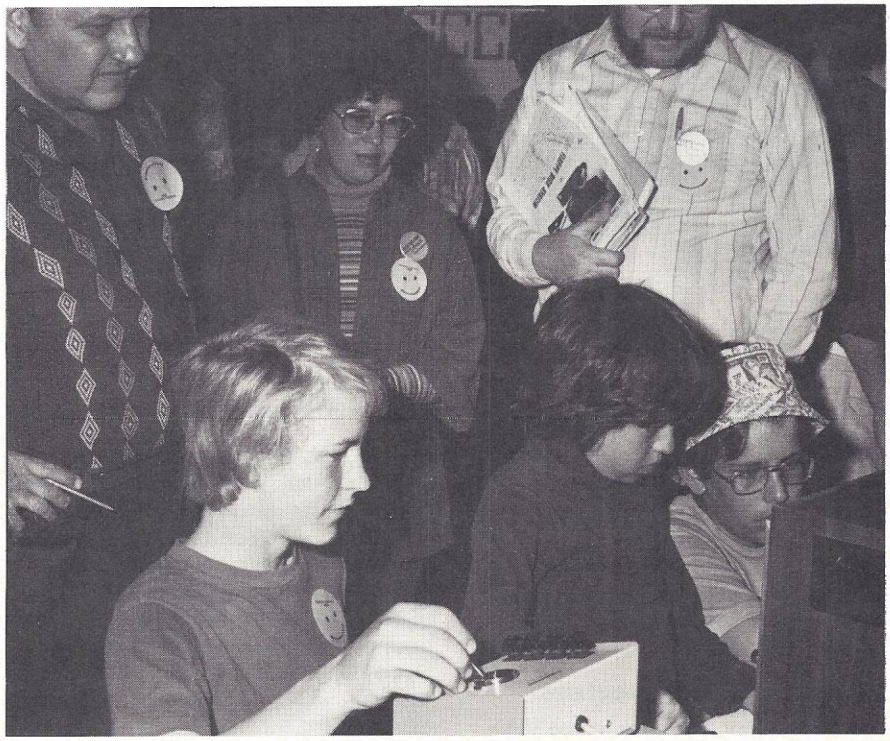
Hobbyists are always seeking more computer for less money. (Who isn't?)

Continued on next page

IMSAI introduced at the show their IMSAI 8048 single board computer. For $299 assembled ($249 in a kit form), this model has a 24 key console, a nine digit calculator-like display, CPU, memory and input/output connections. The single board computer is intended for control applications since it can run any ordinary electrical appliance wired directly into it. IMSAI calls it "the world's first single board control computer." and brags that it can run on a six volt lantern battery.

"The majority of our customers are hobbyists that are buying the system for fun," says Gary Kay of Southwest Technical Products Corp. (SWTPC), a San Antonio electronics firm. Their SWTPC 6800 is turnkey computer that sells for $395 in kit form. The 6800 has never done well in the California area as a whole and we wanted to come out for some exposure. I was surprised when I talked at the 6800 user's meeting. There were quite a few people there. The questions they asked were good ones. I did not expect that kind of response," says Kay.

SWTPC introduced a new $500 video display and typewriter console called the CT 64 Terminal System. It displays 32 or 64 characters on a line and can scroll. (This means that new data entered at the bottom of the video screen causes old data to roll off the top of the screen.) They had hoped to show their new minidisk system but it was not quite ready. It will be out in production quantities in three months, Kay says. SWTPC's other new product was a $40 joystick which can move characters around on a video screen.



Youngsters play with Cromemco T.V. Dazzler. Manipulation of joystick produces brilliant patterns of color on television monitor.

computer store packages

COMPAL, the Byte Shops and the Computer Shack exhibited their special expertise at presenting complete computer systems at low cost. The COMPAL 80 is a $2300 system that includes a microprocessor, 16K of memory, a keyboard, video display and all the necessary interconnecting hardware and software.

"Byte Shops are trying to be diverse places," says Guruprem Singh Khalsa, manager of the Pasadena, CA Byte Shop. "We are trying to have a lot of everything. We're trying to have something for the hobbyist, something for the absolute novice and something for the business person." Byt-8 a system sold at the franchise Byte Shops, is the "lowest cost way for a person to get into the S-100 computer systems. It's $249 and you can plug any of the S-100 boards in it that you want," says Khalsa. The Byt-8 is really an almost empty box with a power supply, a couple of switches and a motherboard. You pick boards you like, plug them in, and create a custom computer of your own.

the hobbyist in business

"I think the name hobby computer is very misleading," says Herb Waite of PerSci. "I don't think I've met what I would consider a pure computer hobbyist." Everywhere at the Personal Computing Show, visitors and exhibitors alike stressed business applications for the personal computer.

"Our software division is here showing the business systems software. That place has been crowded ever since we got here," says Chuck Olsen of Mits.

Continued on p. 114

(Above) — Just part of the crowd at the Personal Computing Show in Los Angeles. United States Robotic Society exhibit in the background.

(Below) — Computer Power & Light put on the biggest display at the show. Besides the double booth shown here, the company sponsored a very popular computer school.

"We offer programs in accounting, general ledger, word processing, whatever you want." Mits was recently bought by Pertec, a giant manufacturer of equipment for all sizes of computers.

The Byte Shop people, The Digital Group, and others, had file editing or file management software systems intended for business applications where a lot of data needs to be processed. Applied Data Communications (ADC) of Santa Ana, CA is not in the hobby market at all and was at the show with their Series 70. "This is a commercial system with a 10 megabyte intelligent minidisk and the only one in its class in the show. Still, the reception was fantastic," says Roger Saville of ADC. The Series has a commercial price too, since the "Bare-Bones I" version starts at $8000, pretty steep for a hobbyist.

Michael Jacoby, an L.A. area dentist, is one hobbyist who found a perfect marriage between his computer hobby and his business. Dr. Jacoby brought his complete audio visual teaching machine to the show. "The computer is running the slide machine and printing the text of what's on the slides," explains Jacoby. The whole system is about $3500 and has a variety of uses in Jacoby's dental office, including patient education. "There are so many uses for this thing in my office, it's unbelievable. I can sit here and dream them up by the millions," says Jacoby.

music for personal computers

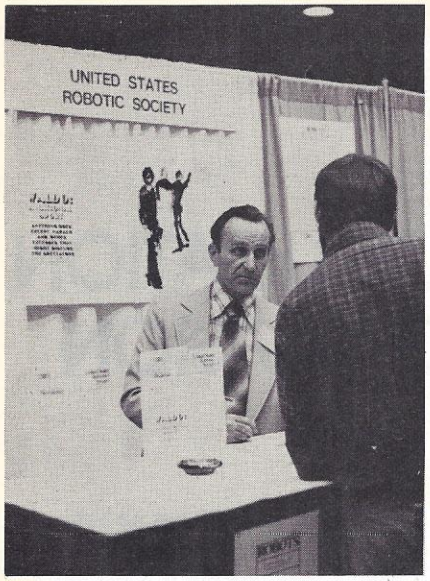
For music or stereo freaks, who happen to be acquainted with an S-100 computer, Galaxy Systems, Woodland Hills, CA, has just the gadget. Their brand new MG-1 Music Generator will turn your home computer into a programmable electronic organ you can listen to directly from stereo headphones or plug into your home amplifier system. It is $299 fully assembled.

The makers claim that no particular musical talent is required to operate the Music Generator. "Take a person who knows nothing about music, I know nothing about music. I can't play and have never had any musical training," explains Bob Chase from Galaxy Systems. "But I can sit down and by following the instructions in the manual I can code up a song.

"The board can generate 49 different notes simultaneously or in any combination for any duration of time. The board includes a 2 watt stereo amplifier so if you want to use it with headphones or modest stereo speakers, you don't need anything else. Otherwise you can run it through a standard stereo amplifier and speakers," says Chase.

Support software for programming music is supplied with the music board. Demonstration programs that can be run immediately without knowledge of programming are also supplied. Ef-

It was a long, tiring day for many of the exhibitors who showed their products to over 3,000 attendees.



Glenn Norris seems downright scandalized at the suggestion his visitor is making for what might be done with robots . . . or maybe he's just exhausted from two days of duty behind that counter . . . and his oft repeated explanation that he's not a robot himself.

forts have been made to keep the programming terminology consistent with organ terminology.

something for everyone

If computer stuff from megabyte machines to music generators still did not strike a visitor's fancy, the Personal Computing Show had other attractions. Hundreds of dollars of prizes were given away. Glenn Norris, president of the United States Robotics Society, was on hand to promote Waldo competition. (See March/April PERSONAL COMPUTING). E&L Instruments showed solderless breadboarding components and microcomputer training devices. Synetic Designs displayed their new intelligent disk storage system. Curtis Electronic Devices had a system that converted Ham radio Morse Code into words and vice versa. Jade electronics offered show discounts for electronic components.

The show drew a wide variety of people, students, computer professionals, housewives, lawyers, doctors. John Brown of E&L Instruments comments: "We talked to a wide variety of people from the young teenage hobbyist to the senior professional computer person who was becoming a hobbyist and every kind of people between them."

PERSONAL COMPUTING plans a Chicago Show, October 27-29. Previously announced show for Philadelphia was postponed because of scheduling conflicts with the National Computer Conference (NCC) and its special Personal Computing section. Boston University will host a summer show in Boston, August 4-6.

MICROTREK

A Personal Computing Book Excerpt

Walt Weller and his associates have been producing first-rate meat-and-potatoes instructional materials in the computer field for some years. This chapter of their latest book is a fair sample of good-humored technical education. If you're already a computer fan hoping to polish up your skill in putting an 8080 to work, this excerpt will help you directly. If you're still an outsider, curious about what these computer people do, the article may tell you more than you really want to know, but you'll gain a sense of what the nitty-gritty is like.

6/MULTIPLICATION AND DIVISION

Like many other small computers the 8080 has no hardware facilities for multiplication and division. These functions need to be performed by the program. It is the purpose of this chapter to develop the necessary methods.

Perhaps the simplest method of multiplying is successive addition. This method consists of using one number as a count to control the number of times the other number is added to a prezeroed register. It is a terribly slow method but it will work, and the speed is not a problem in situations in which time is not critical. We illustrate it in example 6-1 in order to show some of the mechanics involved which will be useful later.

EXAMPLE 6-1

Two positive numbers are located at **DICK** and **SPIRO**. Find their product by successive addition, leaving the product in the H and L registers. First, the product of two N bit numbers will in general be 2N bits long so 2 registers will be necessary to hold it. The addition must then be done in double precision. The easiest way to do this is with the **DAD** instruction, as shown below.

label	*inst.*	*operand*	
	LDA	**DICK**	multiplier to A
	MOV	**C,A**	**DICK** to C register
	ZAR		clear A
	MOV	**B,A**	zero high order multiplier
	LXI	**H,0**	clear both H and L to receive product
	LDA	**SPIRO**	multiplicand to A
SUM	**DAD**	**B**	add multiplier to product
	DCR	**A**	decrement multiplicand
	JNZ	**SUM**	A not zero yet, do it again
	. . .		final product is in H and L

The method shown in 6-1 will not work if the multiplicand in A is zero (why?), and it is inefficient in any case, since it may entail up to 255 repetitions before the product is found.

A more efficient means of multiplication can be had by understanding the multiplication process itself in a little more depth. When pencil and paper decimal multiplication is performed, one number is written under the other, like this:

```
42
15
```

the meaning of this being 42 times 5 plus 42 times 10. When the successive lines of the partial product are written each line after the first is indented to the left to reflect the successively higher powers of ten represented by the successive lines, i.e.:

```
 42   multiplicand
 15   multiplier
210   42 times units digit (5)
42    42 times tens digit. This line is indented because
      the 1 occupies the tens position.
---
630   sum of partial products.
```

Binary multiplication can be carried out in a similar way, with indentation of successive lines representing the successive higher powers of 2 of the multiplier digits. Binary multiplication is a good deal simpler than decimal since no actual multiplication is ever necessary. If the multiplier digit is a 1 the multiplicand is copied into the partial product line. If the multiplier digit is a zero the partial product line is a zero. There are no other possibilities. The process can be carried out just like decimal multiplication. If the multiplier is 5 and the multiplicand a 7, i.e.:

```
   111  multiplicand
   101  multiplier
   111  first partial product line is 7 because
        multiplier units digit is 1.
  000   second partial product line is 0 because
        multiplier 2's digit is 0.
 111    third partial product line is 7 because
 ______ 4's digit of multiplier is a 1.
100011  sum of partial products, final product
```

The sum of the partial products is 32 + 2 + 1 = 35, the correct product of 7 and 5. The method is simple enough. It is only required to implement it with the instructions of the 8080. In the binary multiplication above the digits of the multiplier were taken one at a time from right to left, units, then twos, then fours. There is nothing necessary about this order. It could have been done from left to right, i.e.:

```
   111
   101
 111    multiplicand times 4's digit
  000   multiplicand times 2's digit
   111  multiplicand times 1's digit
100011  product
```

indenting the first line left and reducing this indentation by one on each succeeding line. Further, the three partial product lines could be eliminated. A single partial product could be kept and shifted left one bit before the addition of lower order partial products. This is in fact the way programmed multiplication is done. Beginning with the highest bit, each bit of the multiplier is tested for zero or one. If the bit is a one the multiplicand is added to the partial product. If the bit is a zero the addition is skipped. In either case the partial product is then shifted left one bit and the process repeated as many times as there are bits in the multiplier. Performing this with 8080 instructions is fairly straight forward. The multiplier is kept in A, since this is the only register which allows shifting of a single word. The partial product is kept in H and L. It is double length, since the product will involve more than 8 bits, and this is the only place in which double length addition is possible directly. The multiplicand which must be added to the partial product if the tested multiplier bit is a one is kept in the low order part of either B-C or D-E with the top half cleared, since B-C and D-E offer the only means of adding to H and L directly (DAD instruction).

With this in mind we can proceed to attempt true multiplication, as shown below.

EXAMPLE 6-2

Two eight-bit unsigned integers are located in **GERRY** and **ROCKY**. By the method just outlined, find their product, leaving it in H and L.

label	*inst.*	*operand*	
	LDA	GERRY	multiplicand to A
	MOV	C,A	save as addend in C
	MVI	B,0	clear upper addend
	LXI	H,0	zero to product area in H and L
	MVI	D,8	multiplier bit count to D
	LDA	ROCKY	multiplier to A
MPY	DAD	H	shift product left
	RLC		multiplier bit to carry
	JNC	COUNT	skip if bit was zero
	DAD	B	otherwise add multiplicand to partial product
COUNT	DCR	D	decrement count
	JNZ	MPY	do it again if count not yet zero
	. . .		

At this point the final product is in H and L. It can be stored with an SHLD instruction.

Multiplication of signed numbers involves the same principle as shown above except that the signs must be accounted for. The easiest way to do this is to set both numbers positive, find the product of the positive numbers, and then negate the product if necessary. The sign of a product will be minus if the signs of multiplier and multiplicand are different and plus if they are the same. This is exactly the function performed by the Exclusive OR function described in chapter 1. The sign of the result of exclusive ORing two numbers is the same if they had been multiplied.

The negating of the product requires a double precision operation that has not yet been discussed. The 8080 has an instruction for performing the one's complement operation on the A register, i.e:

CMA

will invert the status of every bit in the A register. 8080 arithmetic however is performed in two's complement, which requires that the number be one's complemented and then incremented to form the negative. This is done by the sequence:

```
CMA        form one's complement
INR   A    and then increment
```

The assembler provides a pseudo-op for this pair of instructions, namely:

TCA

which will assemble as if CMA and INR A had been written.

The assembler pseudo-op is not of help for double precision numbers, however. It is easy enough to form the ones complement of both parts of a number, but increment-

ing it requires that any carry out of the low half be propagated into the high half. This is a special case of double precision addition, shown in example 6-3.

EXAMPLE 6-3

A double precision number is in the H and L registers. Negate the number, leaving the result in H and L.

```
label   inst.   operand
        MOV     A,L       low half of number to A
        TCA               twos complement low half
        MOV     L,A       and return it to L
        MOV     A,H       high half to A
        CMA               ones complement high
                          half. At this point the
                          flags are still set to the
                          values they had after the
                          TCA, i.e., CMA + INR A,
                          If the INR resulted in a
                          zero a carry into the
                          high half is required.
        JNZ     T2        check for zero
        INR     A         increment high half if
                          low half was zero
T2      MOV     H,A       and return high half to H
        . . .
```

A note of caution about negation is in order here. The reader will recall that there is one more negative number than positive in a two's complement system. In eight bits the maximum positive number that can be expressed is +127, while the maximum negative number is −128. This usually is not a source of trouble, but if an attempt is made to negate the maximum negative number an invalid result is obtained i.e.:

```
 10000000  =  -128 (base 10)
 01111111     the one's complement
       +1     the increment
 --------
 10000000     the same number back!!
```

In a computer equipped with an overflow flag this condition would turn the flag on. Since the 8080 has no such flag the programmer is obliged to check for this condition if the numbers in the problem would allow it to arise. If the sign after negating agrees with the sign before negating the number cannot be properly negated, i.e., it is the maximum negative number for the length being used. The ability to sense this condition automatically is one of the advantages of a computer with direct overflow sensing.

The method of multiplication just shown is faster than successive addition but still slow. In the case of a special constant multiplier there is a shortcut that can be taken. Suppose it were required at some point to multiply by pi (3.14159 . . .), a number whose value will not change from one execution of the program to the next. A fairly good approximation to pi is 201/64. Since the denominator of this fraction is an integral power of two, the division can be eliminated in binary, it amounts only to a shift of 6 bits to the right. A number can be multiplied by pi therefore by multiplying by 201 and shifting the product 6 places right. This is a very useful trick for special fixed multipliers. Another useful ratio is 87/32 for e, the base of natural logarithms. This multiplication is performed by multiplying by 87 and then shifting right 5 places, effectively dividing the product by 32. Some such ratio can be found for any special multiplier in a few minutes with a hand calculator. A table of these special ratios is given on page 109 of the author's earlier *Assembly Level Programming for Small Computers* (Lexington, 1975). Finding the ratio is fairly easy. Suppose it were necessary to multiply by 38.6 at some point in the program. Multiply the 38.6 by a power of two, say 16. Round the result to the nearest integer and divide by 16. This quotient will be the actual value of the multiplier. The rounded result of the multiplication above is the numerator of the sought after fraction, 16 being the denominator. Thereafter to multiply by 38.6 simply multiply by the numerator, 618 in this case, and shift the result right 4 bits, effectively dividing by 16. The effective multiplier in this case is 38.625, a good approximation. If better precision than this is required use a larger denominator, 32 or 64, and repeat the process.

Like multiplication by successive addition, division can be accomplished by successive subtraction. Also like multiplication by successive addition, division by successive subtraction is a slow and inefficient method. To understand division on a binary computer we again detour back to the pencil and paper decimal case. Consider the division:

```
23 )417
```

the 23 being called the *divisor* and the 417 being known as the *dividend.* Such a division is begun on paper by selecting a number of leading dividend digits such that the divisor 23 can be divided into these digits without producing a quotient greater than 9. In this case the chosen digits are 41. The 41 is divided by the 23, which yields a quotient of 1 and a remainder of 18. This is usually written:

```
      1
23 )417
    23
    18
```

The remaining effective dividend is 187, which is indicated by "bringing down" the 7 and appending it to the 18:

```
      1
23 )417
    23
    187
```

This new dividend, 187 is then divided by 23 for the next quotient digit:

```
     18
23 )417
    23
    187
    184
      3
```

This final 3, the number left after all possible multiples of the divisor have been removed from the dividend, is called the *remainder*, the number of times the divisor can be subtracted from the dividend being called the *quotient.*

Essentially the same method is used in division on a binary computer. The only differences are restraints imposed by the structure of the computer itself. Division is conventionally accomplished on computers in such a way that the quotient is entirely containable in one computer word. This restraint places conditions on the relative sizes of dividend and divisor. A condition in which the relative sizes of divisor or dividend would result in a quotient more than one computer word long is known as a *divide check* or *divide fault* condition. Computers which have hardware multiplication sense this condition and indicate it by turning on overflow and (usually) aborting the divide. For signed 8 bit numbers, i.e., 7 magnitude bits plus sign in the 8080, the con-

dition is that if the rightmost 7 bits of the dividend are excluded, the remaining dividend bits must form a magnitude *less* than that of the divisor. Consider the division:

```
000010101 1110111   dividend
 00010110           divisor
```

The above division is legal because the condition is met. Splitting off the rightmost 7 bits (indicated by the gap in the dividend), the remaining bits to the left of the gap form a magnitude less than the divisor. The division:

```
000010101   1110111   dividend
 00010101
```

is *not* legal because the divisor and the high dividend bits are equal. This constitutes the divide fault condition. It is easy to verify that this is true for the above divide. The decimal value of the dividend 0000101011110111_2 is 2807_{10}. The decimal value of the divisor is 21. The decimal quotient is 133 and the remainder 14. The number 133_{10} is too large to be held in a signed word, the largest possible number being +127.

The actual process of division involves shifting and subtracting only. It is never necessary to guess how many times the divisor will go into the available dividend bits. If the divide fault condition is not present we always begin with the high dividend less than the divisor. The division is begun by shifting the entire double length dividend left one bit and attempting to subtract the divisor from its high 8 bits. If the result was negative the divisor is added back in and the high quotient bit is a zero. If it was positive the high quotient bit is a 1. This process is repeated as many times as there are bits in the divisor, this number being the same as the number of low dividend bits. The double length left shifts correspond to the "bringing down" of the digits in paper division. As a matter of convenience, the generated quotient bits are introduced at the low end of the dividend which is being shifted up. In this way the quotient is automatically accounted for and appears in the place of the low dividend when the division is finished. The remaining portion of the dividend, that which is left after all possible subtractions have taken place, is left in the old high dividend place and forms the remainder. This division is shown in example 6-4.

EXAMPLE 6-4

A positive double precision dividend is in H and L and a single precision positive divisor is in C. Divide H and L by C, leaving the quotient in L and the remainder in H. Check for the divide fault condition and jump to location **DFAULT** if it occurs.

```
label   inst.   operand
        MOV     A,C       divisor to A register
        ORA     A         check for zero divisor
        JZ      DFAULT    zero divisor, fault
DIV     MVI     B,7       magnitude bit count to B
        DAD     H         shift dividend left
        MOV     A,H       check high magnitude
                          bit
        ORA     A
        JM      DFAULT    if the highest magnitude
                          bit was on the divide is
                          impossible, since the div-
                          idend has 15 significant
                          magnitude bits. A 15 bit
                          dividend divided by a 7
                          bit divisor will yield a
                          quotient with 8 magni-
                          tude bits, i.e., too big.
        MOV     A,C       fetch divisor to A
        CMP     H         compare divisor to high
                          dividend
        JAL     DFAULT    A is less than high divi-
                          sor, divide fault
        JEQ     DFAULT    A equals high divisor,
                          divide fault
DIV2    DAD     H         shift dividend left
        MOV     A,H       high dividend to A
        SUB     C         subtract divisor from
                          high dividend
        JM      DIV3      quotient bit zero if
                          negative
        INX     H         quotient bit one, in-
                          crement vacated position
        MOV     H,A       and set new high dividend
DIV3    DCR     B         decrement count
        JNZ     DIV2
        ...
```

At this point the quotient is in L and the remainder is in H. If two cells are defined:

```
QUOT  ***
REM   ***
```

both quotient and remainder can be stored with a single **SHLD QUOT** instruction. Note that **SHLD** stores L first and then H in the next higher memory location.

It is sometimes necessary to perform multiplication and division in multiple precision. Multiplication is fairly easy to do. Consider the double precision multiplier to be represented as A + B, with A the high order part, and the double precision multiplicand to be represented as C + D, with C the high order part. The product is then:

$$(A + B)(C + D) = AC + AD + BC + BD$$

Each of these separate single precision multiplications yields a double precision product. The separate products are then added in multiple precision according to the following scheme:

```
ACH   ACL
      ADH   ADL
      BCH   BCL
            BDH   BDL
```

the suffixed H meaning high order and L meaning low order. The product of two 16 bit numbers will be 32 bits long. If the numbers were integers the entire 32 bits must be retained. If the numbers were fractions with 16-bit precision, then only the first and second orders, counting from the left need be retained, though the programmer may wish to use the high bit of the third order as a rounding bit for the second order. If you do this, remember that incrementing the second order may result in a carry into the high order. This carry *must* be checked for and propagated, just as it was propagated during double precision negation.

Division of one double precision number by another is usually done in the context of fractions, i.e., both dividend and divisor are considered to be pure fraction. In this case a quotient is developed and the remainder is ignored. This is usually done by an approximation. Consider a double precision dividend A + B and a double precision divisor C + D, with A and C being the high order terms. The result of the

division looks like this:

$$\frac{A+B}{C+D} = \frac{A+B}{C} - \frac{DA}{C^2} - \frac{DB}{C^2} \ldots$$

the series of terms continuing. Given the relation between the high and low order terms, it is fairly easy to see subjectively that this series will converge rapidly. DA involves the product of a first and second order term and is thus second order. DB is the product of two second order terms and is thus a fourth order term. For all but the most demanding applications, it is sufficient to use:

$$\frac{A+B}{C} - \frac{DA}{C^2}$$

which will yield a result good to 15 bits, the 16th bit being uncertain, and tending to be high because the first neglected term is negative. If you attempt to implement this remember while programming the second term that since DA/C is guaranteed to be divisible if there is no divide fault condition, there is no guarantee that this quotient can be further divided by C (to achieve division by C^2) without divide fault. After the first division, shift the resulting quotient right one bit before dividing it by C again. This guarantees divisibility. The quotient must, of course, be shifted left one bit to compensate. After that add all the terms together, but be careful that the sum does not overflow. Since the approximation tends to be high in the last bit this can happen. Check for this case and set the final result to 0111111111111111, the best approximation.

As with multiplication, there are short methods of dividing by fixed divisors. If the divisor is a power of two the division can be done by simple shifting. Division by three can be approximated by shifting and adding alternate powers of two, like this:

a) 00111010 number to be divided by 3, a 58_{10}
b) 00001110 shift right two bits and save
c) 00000011 again, and save
00000000 number is now zero, stop

adding a, b and c above, 58 + 14 + 3 = 75 or in binary

01001011

Shifting this right two bits yields

00010010

the last bit shifted out being a 1. The 1 will still be in carry, so it can be used to round the result, giving $00010011_2 = 19_{10}$, the correct integer quotient for 58/3. This does involve approximation and the answer will not in general be as precise as that obtained by a full divide but it is *far* faster than a full divide. The above method is based on the fact that:

$$N/3 = N/4 + N/16 + N/64 \ldots$$

the series continuing infinitely. The denominators of all the terms are integral powers of two, so they can all be determined by simple shifts.

The initial shift of the series (N/4) is deferred until last to preserve precision.

Other useful special division series are:

$$N/5 = N/8 + N/16 + N/128 + N/256 \ldots$$
$$N/7 = N/8 + N/64 + N/512 + N/4096 \ldots$$

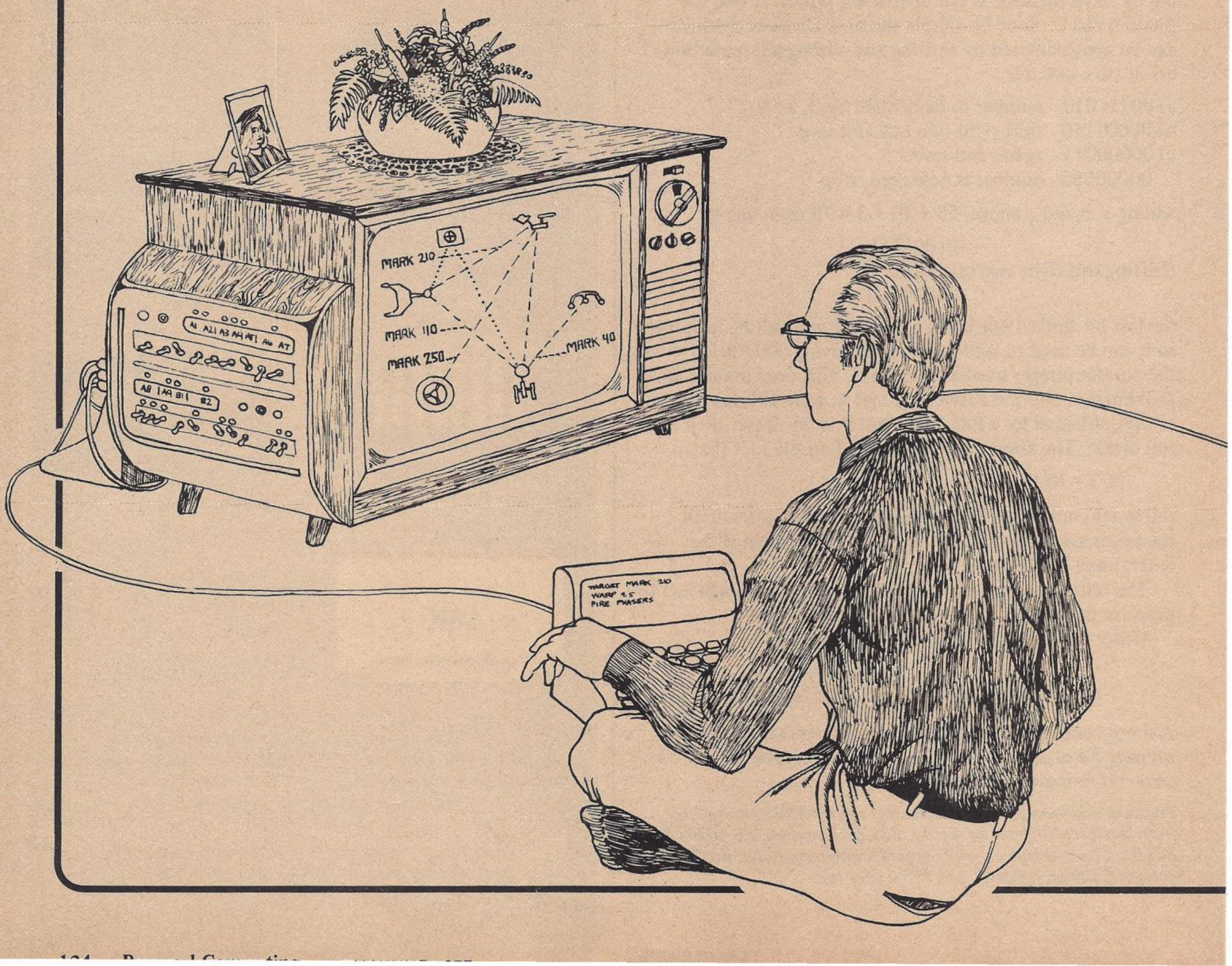
A more complete table of these special series can be found on page 88 of *Assembly Level Programming for Small Computers* (Lexington, 1975)

Practical Microcomputer Programming: the Intel 8080 *is available from Northern Technology Books, Box 62, Evanston, Ill. 60204 for $21.95. Check or money order should accompany order; no COD's accepted. Illinois residents add $1.10 sales tax.*

Top-Down Design of

People play elaborate games with their personal computers. Manufacturers furiously develop and market "video games" that make increasing use of computer logic. The computer and the video game seem destined to meet in the average man's living room. Scott Guthery, president of Computer Recreations, Inc. in East Windsor, N.J., offers here a basic approach to writing genuinely new games, to creating a body of literature for this new medium we have developed

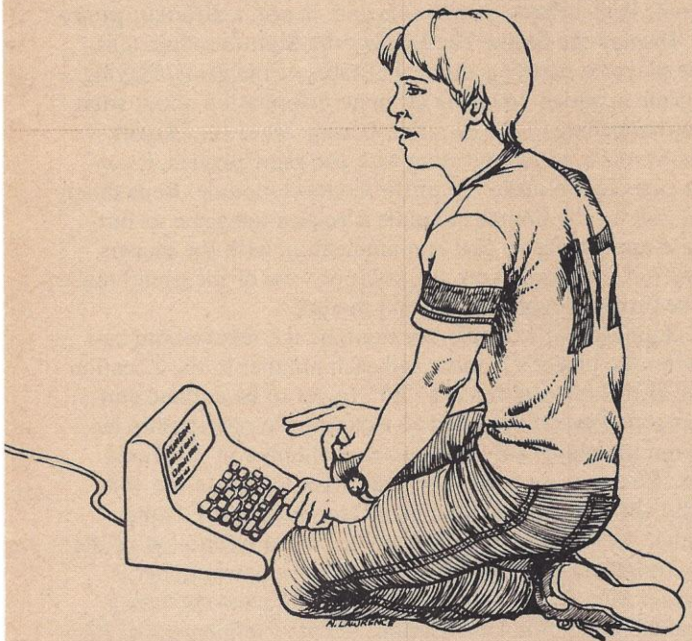
by Scott B. Guthery



Computer Games

Games are customarily described in terms of the paraphernalia used to play them for example, in chess we speak of a piece called a bishop which moves on the board along diagonals and in cribbage we talk about pegging out. Yet pieces, boards and pegs are not parts of the games themselves but simply parts of the tools used to keep track of them. Games and their accounting tools are so intimately interwoven that we often tend to regard them as one and the same thing. A moment's reflection will convince you they are not. There is no reason why we couldn't describe a method for scoring cribbage that uses chess pieces and a chess board or a method for playing chess using pegs and cribbage boards. Once we recognize that a game and its accounting machinery are quite separate and distinct, we are ready to think seriously about designing and building computer games.

In a computer game, the computer becomes the accounting machine. There is no longer any need for concepts like pieces, boards, dice, cards or any of the other accouterments of non-computer game accountancy. Rather than providing game players with bookkeeping materials such as these and exhaustive rules for their manipulation, the computer game designer implements the entire game accounting mechanism as a computer program. The game designer thus frees the game players from the drudgery of keeping track of the game and allows them to concentrate on the strategy and tactics of playing the game. This simple shift of responsibility of game tracking from game player to game designer will, I predict, cause a large change in the kinds of games we will be playing in the future and how we think about games in general.



We will consider the fundamentals of computer game design, trying to answer the question: what are some possibilities for the elementary processes of a computer game? We start only with the conviction that computer games are something quite different from computerizations of existing game mechanisms.

Design Overview

The key precepts of our design strategy are (1) start with a specification of what you want to do and continually refine this specification, defining as much as possible of the interface to the outside world and adding more detail, until the design is as precise as possible without being code (this procedure is called top-down design) and (2) modularize wherever natural and with an eye toward minimizing the amount of information flowing between modules (this procedure is called information hiding). The intent of this design strategy is to allow us the greatest possible freedom for experimentation and redesign. The result is that we will be proceeding through a series of step-by-step refinements from the vague to the specific while at the same time trying to uncover some basic processes of computer gaming.

Since we seem to learn best by doing, we will design a family of computer games. Keep in mind that what we will be building is only an example, an illustration of how one might go about constructing a computer game. To begin with, the computer game model we will be discussing is defined in terms of three main modules: Context, Algorithm and Data. You may want to begin with another overall modularization. The purpose of this article is to help you think about doing this and how to proceed next.

Context module The Context module contains all the information which determines the outward appearance of the game. That is, is it a war game, a real estate game, a personality game, a sports game, etc? The Context of a game is, in some sense, the link between the game's reality and the player's reality. It is the primary source of the "feel" of the computer game. Only the Context module knows what the game is "about." This information is hidden from the Algorithm and Data modules and, hence, can be changed freely.

Algorithm module The next main module is the Algorithm.

This module controls the running of the game. It determines what happens, when it happens and how it happens. What the Algorithm module does – in other words, the events of the game and their sequencing – is hidden from the Context and Data modules. This seems reasonable since the Algorithm is, in some sense, the dynamics of the game while the Context and Data are the statics – external and internal, respectively – of the game.

Data module Finally, the records that the Algorithm uses to run the game are maintained by the Data module. The Data module replaces the pieces, boards, pegs, cards, etc. of non-computer game technology.

Module integration These three basic game modules are arranged sandwich style with the Algorithm being the meat: When the Algorithm wants to interrogate or change the state of the game, it communicates with the Data module. When the Algorithm wants to interact with a game players, it communicates through the Context module. Our next task is to specify the form and content of these communications. Following our top-down design strategy, we will postpone saying anything about what goes on within any of the main modules. We will be concerned only with what the communication between the modules looks like.

Data Module Communication

The status of any player in our family of games will be communicated to and from the Data module as two numbers, A and B. These two numbers don't have any intrinsic meaning or interpretation. The meaning is supplied by whatever use is made of them. We will be able to retrieve these numbers using the subroutine PUT. Thus, for example, if somewhere in the Algorithm we wanted to use the current A and B values for player 2, we would execute the subroutine call GET(2,A,B). If we wanted to set the current A and B values for player 1 to 6 and 9 respectively we would execute PUT(1,6,9). Notice that we have not specified how the Data module handles the GET and PUT calls on it. All we have specified is what the Data module looks like to its "outside world" – the Algorithm.

Context Module Communication

The Context module appears as two lists composed of individual items which contain a name and a number. One list, called the Location list, might contain items like CLEVELAND, OHIO 50.2 or MARS 1834.53. The other list, called the Thing list, might contain items like 1957 CHEVROLET 6.2 or EGG MCMUFFIN 134.5. The items in each list are numbered and are accessed using these numbers. Thus, for example, GETLOC (I,LS,M) returns the name of the Ith entry in the Location list in LS and its number in M. GET–TING(I,TS,N) returns the name of the Ith thing in TS and its associated value in N. Since we may need to know how many entries there are in each list, LOCS(K) and TNGS(L) return the number of locations and things in K and L respectively. Notice, once again, that we have specified only what the Context module looks like to the Algorithm. We have put off until later exactly how the Context module maintains its lists.

The Algorithm

The Algorithm module, as we have said previously, is where the action is. It manages the overall flow of the game and implements the sequence of events which define the game. Our Algorithm module is defined in terms of five submodules. Let's consider each of the Algorithm's submodules in turn with the intention of refining them and specifying how they interact with the Data and Context modules.

Pick-a-player If we assume N people are playing the game, the Pick-a-Player module must decide which of these players may alter the state of the game next. (Note that one property of this Algorithm module is that the players play one at a time. This may not be the case for some other Algorithm module.) We are going to assume that the Pick-a-Player module looks over the A and B Data values for each player, computes some number from these two numbers – call it Pick(A,B) – and picks the player with the smallest value of this number to go next. Therefore, our Pick-a-Player submodule looks like

```
PICK-A-PLAYER

the player with
 the smallest
   Pick (A,B)
   goes next
```

Some pidgin Basic program code which implements this Pick-a-Player is:

```
        P=0
        M=9999
        FOR I=1 TO N
          GET (I,A,B)
          V=Pick (A,B)
          IF V>M THEN 100
          M=V
          P=I
100     NEXT I
```

Now, the Pick-a-Player submodule of the Algorithm module is completely specified except for the Pick function. This is the bottom of our Top-Down design. Possibilities for Pick include Pick(A,B)=A+B Pick(A,B)=SIN(A*B) or PICK(A,B)=SQRT(A)*RND*B. For each choice of Pick, we get a different Pick-a-Player submodule and, hence, a different game.

Display-the-Status The Display-the-Status module tells the player something about the status of the game. Playing a game in which you were given no information about what was happening might be rather boring. Since the Display-the-Status is communicating with the game players, it can be expected to make use of the Context module. Remember, the role of the Context module is to give the game its outward appearance so that communication with the players (the outside world from the point of view of the game) must take place through the Context module.

Suppose, for example, we examine the relationship between the player's A value and each number in the Location list. If this relationship is a "hit" (a yet to be defined concept) then we will proceed to examine the relationship between the player's B value and each number in the Thing list. Whenever we get a "hit" on this second pass, we will print the current Location name and the current Thing name. An easy way to define a "hit" for a refunction of the two numbers under consideration. If this new number is greater than 0 then we have a "hit"; otherwise we have a "miss." Using this specification, the next refinement of the Display-the-Status submodule is:

```
  Display-the-Status

Print LOCATION name
and THING name when

At (A,LOCATION number)
        and
 Has (B, THING number)

are greater than zero
```

As with the Pick-a-Player module, we can easily write down some pidgin Basic code which implements Display-the-Status and leave the definition of the At and Has functions to the very last.

```
       GET (P,A,B)
       LOCS (K)
       TNGS (L)
       FOR I=1 TO K
         GETLOC (I,L$,M)
         X=At (A,M)
         IF X<= 0 THEN 100
         FOR J=1 TO L
           GETTHG (J,T$,N)
           X=Has (B,N)
           IF X<= 0 THEN 50
           PRINT 'Player ';P;' has a ';T$;' at ';L$
 50      NEXT J
100    NEXT I
```

The simplest definitions for At and Has generate a "hit" whenever A or B are sufficiently close to Location number and Thing number respectively. Thus, we could let At (A, number)=P1-ABS(A-number) and Has(B,number)=P2-ABS (B-number) and then just set the parameters P1 and P2 to complete the specification of the Display-the-Status module. Each setting of P1 and P2 generates a new computer game and, of course, different choices for the At and Has functions define new games.

Since it is usually in the Get-a-Command submodule that the "rules" of the game are imposed on the players, in the interest of simplicity, expediency and experimentation, we are simply going to have the player input a number:

```
GET-A-COMMAND

The current
player inputs a
number
```

which is immediately implemented with the pidgin basic code:

```
INPUT C
```

In the spirit of the Pick-a-Player and Display-the-Status submodules, one could easily limit what values the player may input using his current A and B values, or one could have the player pick a Thing or a Location from a list of Things and Locations computed using his A and B values. Whatever one does in Get-A-Command, our computer game design calls for the passing of a number C to the Update-the-Game submodule. Notice once again how we have confined the interaction between the various modules to the communication between them. That is, the relationship of any module to all the others is defined only in terms of the values it must be supplied with upon entry and the values it returns. How it gets between these input values and these output values is totally hidden from the outside world. It is this design strategy which makes the generation of new computer games from old a trivial task.

Update-the-Game The Update-the-Game submodule simply computes new A and B values for the current player given the current A and B values and the output of the Get-a-Command module, C:

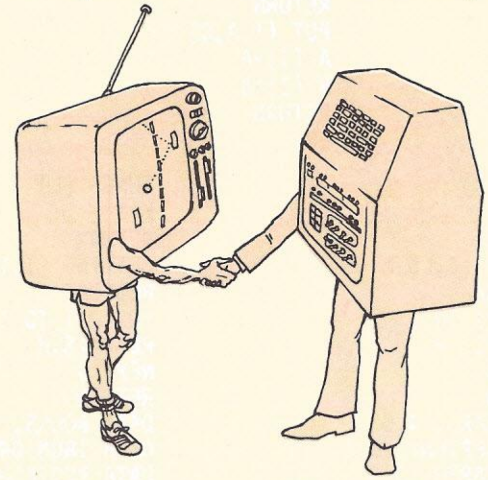
```
UPDATE-THE-GAME

A=NewA (A,B,C)
     and
B=NewB (A,B,C)
```

The pidgin BASIC code for this module is simply,

```
GET (P,A,B)
A=NewA (A,B,C)
B=NewB (A,B,C)
PUT (P,A,B)
```

The choice of the NewA and NewB functions is, of course, critical to the playability of the game but *not* to its implementation of coding. Some NewA and NewB functions will result in reasonable games and some in totally unreasonable games. But the game designer's attention is now focused on the play of the game, not on the mechanics defining it.



Check-for-End Every game must have an end and a winner. Computer games are no exception. After each Update-the-Game, we will determine from each Location and each Thing a score based on the NewA value and the NewB value. If the total of these scores is greater than a fixed value, the current player will be declared the winner.

```
      CHECK-FOR-END

     T is the sum over
     all LOCATIONs and
       THINGs of

         Score

     T< E      T >= E

     Go To     Winner
      Pick       is
       a         P
     Player
```

Some pidgin Basic code which implements Check-for-End is

```
GET (P,A,B)
LOCS (K)
TNGS (L)
T=0
  FOR I=1 TO K
  GETLOC (I,L$,M)
  FOR J=1 TO L
    GETTNG (J,T$,N)
    T=T+Score (A,B,M,N)
NEXT J
NEXT I
IF T< E THEN GO TO Pick-a-Player
PRINT P;' WINS'
STOP
```

Obviously, the Score function must be chosen with an eye toward the previous definition of the At, Has, NewA and NewB functions. However, any choice whatever along with a choice of the parameter E yields a perfectly well defined Check-for-End module. Whether the game does in fact ever end is another problem altogether.

Inside the Data and Context Modules

Now that we have completely specified the Algorithm module and have seen how it interfaces to the Data and Context modules, let us return and show how one might go about implementing these two modules. The specification of both modules immediately suggests the use of arrays. Thus, for example, we might have:

```
GET (I,A,B)
DIM A (10), B (10)
A=A (I)
B=B (I)
RETURN
PUT (I,A,B)
A (I)=A
B (I)=B
RETURN
END
```

```
LOCS (K)
K=5
RETURN
GETLOC (I,L$,M)
RESTORE
FOR J=1 TO I
READ L$, M
NEXT J
RETURN
DATA MARS, 4.5
DATA NEPTUNE, 6.1
DATA EARTH, 3.9
DATA VENUS, 7.3
END
```

```
THNGS (L)
L=5
RETURN
GETTING (I,T$,N)
RESTORE
FOR J=1 TO I
READ T$,N
NEXT J
RETURN
DATA BOOKS, 5.7
DATA IRON ORE, 8.2
DATA MEDICINE, 19.3
DATA TOOLS, 8.2
END
```

but nothing in the definition of the interface to the Data or Context module prevents us from doing more interesting things inside them. Thus, for example, we might add some random noise onto the returned numbers or occasionally set the string variables to ***ZAP***. The point is that we can do whatever we like inside these modules as long as we meet the interface specifications.

Realizing the Potential

Many words have been written about the potential for computer games but surprisingly little has been done toward realizing this potential. One reason for this, I suspect, is that we are still in the stage of putting existing game concepts on the computer rather than trying to discover what new concepts are appropriate for the computer gaming environment. Existing games are intimately tied to the paraphernalia used to play them. Most existing computerized games have attempted to automate this paraphernalia rather than replace it.

In this article we have attempted a first exploratory cut at defining computer game primitives and building games from them. While the games we have constructed are of dubious merit in themselves, they do give us a glimpse of how computer games will be different from the games we are used to playing. First, since all the game accounting is taken care of by the machine, not only will computer games have far fewer rules than conventional games but they will be far more complicated. Second, there will be a much greater discovery or information gathering dimension to computer games than in their non-computerized counterparts. Finally, computer games may require a much deeper set of logical and analytical thought processes on the part of their players. All that can really be said with certainty is that computer gaming is a totally new and rich form of gaming whose dimensions have not even been discovered, let alone explored.

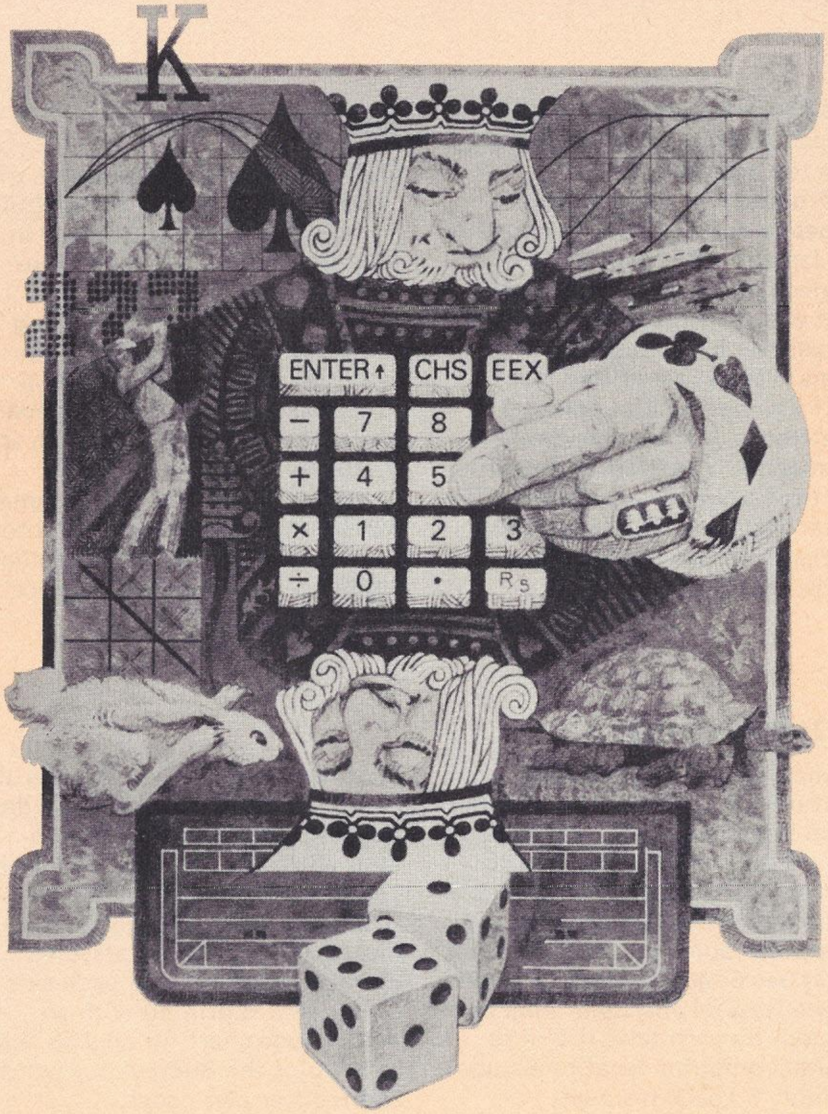
PART 2 random access

Programmable Calculator Games

A large literature of games has developed for pocket calculators, some of them entertaining even to those who don't enjoy mathematical puzzles. Classy programmable calculators can even handle respectable versions of Star Trek.

Hewlett-Packard, manufacturer of the very classy HP-67 programmable calculator has even undertaken serious pursuits to offer Games Pac I, a collection of 19 programs with lots of supporting information — for $35. Caught up in the spirit of the thing, H-P has produced uncommonly nice artwork to illustrate the fun. Their King of Spades looks less villainous than most.

GAMES PAC I will be appearing in the stores any time.



Creeping Control

The British Computer Society has recommended that licenses be required for computer systems dealing with personal data. The proposal is that all such systems, public and private, within the United Kingdom be controlled under licenses that cost from fifty to five hundred pounds per year.

This approach is bound to get the attention of U.S. legislators. Personal computer owners may now want to keep their homes and systems extra tidy for the surprise visits of government inspectors checking their computer applications. Presumably the regulatory agency will insist that no warrants are needed for such inspections and traditionalists will insist that they do. The discussion may prove interesting.

Computer Pictures — 30¢ each

A book called *Artist and Computer* was cheerfully reviewed in our last issue and Russ Walter pointed out that if the price of the book is divided by the number of pictures it offers, the pictures come to three cents each.

Ruth Leavitt, Editor of that book, has written to say that the Computer Arts Society is now offering a set of approximately 135 slides on computer art, suitable for projection before classes and other meetings, for a mere $40.00. That works out to more like thirty cents apiece, and still seems like a bargain. Send orders (and checks) to Ruth Leavitt, 5315 Dupont So., Minneapolis, Minn. 55419.

On the Air

The Computer Program is on the air. A radio talk show for computer hobbyists, it features live guest interviews, call-in questions, product reviews, discussions of books

random access

and hardware, music and all those special gossipy features that entertain the hobbyist.

Producer Richard Gardner hopes to build his first program of 22 January 1977 on WBUR-FM, Boston, into a syndicated series that will reach listeners nationwide. Is radio an idea whose time has come? Tune in and find out.

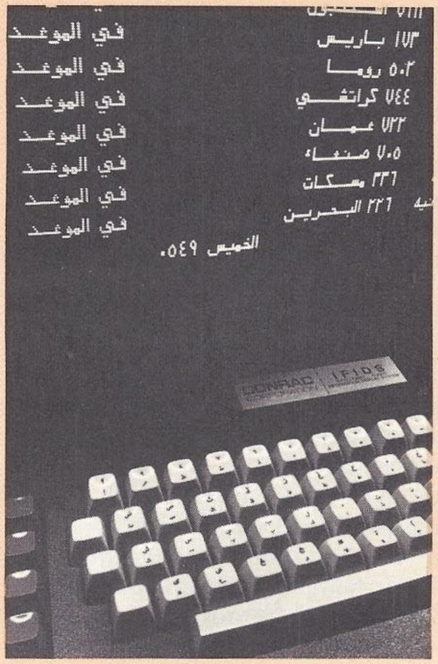
If you have useful material to contribute or a burning desire to arrange for broadcast of the programs in your area, write the Computer Program, P.O. Box 134, Harvard Square, Cambridge, Mass. 02138.

Babel

Quite apart from the problem of "computer language" compatibility, designers are confronting international problems with "human language" compatibility . . . and producing some interesting solutions. Hewlett-Packard, for example, has developed special terminals with modified keyboards for use by speakers of Finnish and Swedish.

Western users of CRT terminals will be entertained by the photograph here showing Arabic calligraphy on a CRT display. Notice also the characters on the keys.

As non-professional computing develops around the world, amateurs have an opportunity to influence communications standards that are not strictly tied to technical concerns. Is anyone actively pursuing this matter?



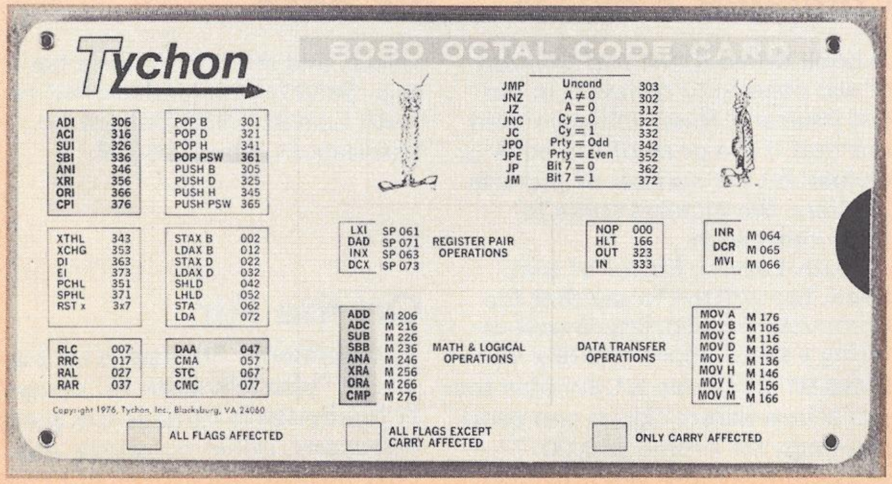
A bilingual Intelligent Flight Information Display System (IFIDS) will provide flight information in Arabic and English on television monitors and split-flap boards located throughout the terminals of eight Saudi Arabian airports. The computer-driven IFIDS, engineered and installed by Conrac Corporation, New York City, is the first orthographic display of Arabic calligraphy in a largescale computerized system. Twin console keyboards and video monitors in Arabic (above) and English permit updating information. Origin and destination cities, airline names and standard IATA remark codes are stored in the computer memory. Changes in flight information using this data are entered on the English keyboard and automatically displayed in both languages.

Backsliding

Some observers have suggested that pocket calculators and sliderules might be built in to the front panels of computer systems so that operators could do quick calculations without disturbing the computer during problem-solving exercises or games like Star Trek. While Keuffel and Esser may be pleased by such talk, it strikes digital computer fanatics as rude humor. To each his own.

Even the digital fans will be glad to see a new product offered by TYCHON, Inc. (P.O. Box 242, Blacksburg, VA 24060). It's an 8080 Octal Code Card, a sliderule-like aid for programming and debugging 8080 software. It contains all the mnemonics and their corresponding octal codes. Instructions are color coded to indicate which flags are affected during execution. The back side of the card is printed with an ASCII code chart for all 128 characters plus the 8080 status word and register pair codes.

Price of the pocket-sized, 6.5" x 3" device is $2.95. A hexadecimal card is promised shortly, too. Handy tool.



Learn About Computers Quickly

The quickest way to learn about computers is to read The Secret Guide to Computers, a series of four booklets introducing all aspects of computers thoroughly. Part 1 covers BASIC; part 2, applications; part 3, other computer languages; part 4, hardware and systems programming.

The pace is quick. The first paragraph says, "10 minutes from now, you'll know how to run a computer. If you have a computer in front of you, it will already be printing answers to your problems."

Scared? "No matter what buttons you press, no matter what commands you give, you won't hurt the computer. The computer is invincible! So go ahead and experiment. If it doesn't like what you say, it will gripe at you, but so what?"

Like a protective mother, the book guards the reader against all difficulties. "It's impossible to hit the SHIFT key and another key at the same time; you'll hit one key before the other. So hold down the SHIFT key first; and while you keep holding it down, give the other key a light tap."

The first program prints 17.9-2, -6+1, and "I LOVE YOU". By showing "Mary Poppins meets Frankenstein" and other weird examples, the first 15 pages cover all the essentials; PRINT, END, math operations, numeric and string variables, INPUT, GO TO, STOP, IF, and how to create, edit, save, and revise programs. The rest of the book digs deeper.

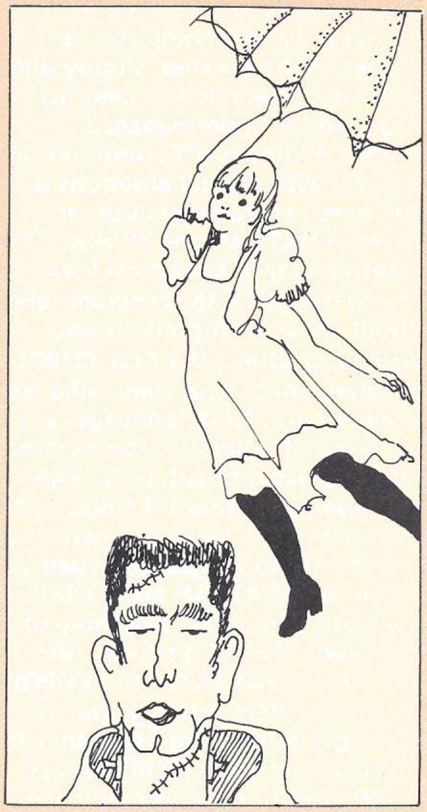
How can a vague idea become a program? First, decide on your ultimate goal. Then simplify it; pick a goal that's less ambitious and more realistic. Draw a picture, find the lines typed by the computer. They become the PRINT statements in your program. Find the lines typed by the human: they become the INPUT statements. Write the PRINT and INPUT statements on paper, with a pencil, and leave blank lines between them. Fill in the blanks later, by asking yourself how you'd get the answer if you didn't have a computer.

- Would you use a mathematical formula? Put it into your program, but remember that the left side of the equation must have just one variable.
- Would you use a memorized list, such as an English-French dictionary, the population of each state, or the weight of each chemical element? That list becomes your DATA, and you need to READ it. If you'll use the list more than once you use subscripts.
- Would your reasoning repeat! If you know how often to repeat, say FOR NEXT. If you're not sure how often, say GO TO or IF THEN. If the repetition occurs only after several other activities have intervened, call the repeated part a "subroutine", put it at the end of your program, and say GO SUB.
- Would you choose among several alternatives? Say IF THEN. To make the computer choose arbitrarily, say IF RND < .5 THEN.
- Would you compare two things (A and B)? Say IF A=B THEN.

Trim your program. "Skim through it and eliminate any lines that are silly, such as a GO TO that goes to the next line; a GO TO that that goes to the END; a GO TO that goes to a GO TO; a THEN that goes to the next line; and THEN that just skips over to a GO TO; and IF followed by an IF that has the opposite condition." Examples show how to correct the silly logic.

The book shows how a sample program that takes a computer several months can be rewritten to take a tiny fraction of a second. The book is chock full of hints on how to handle loops and debug, your masterpiece.

Part 2 of the Secret Guide covers Russian, English, baseball, math, encyclopedias, psychotherapy, brainwaves, poetry, physics, pornography, art, tutoring, medicine, religion, personal relationships, robots, bartenders, doctors, jobs, artificial intelligence, your home,



unemployment, crime and invading your privacy. The treatment is thorough: to analyze the ELIZA program, which makes the computer become a psychologist, you even peek at the program's dictionary.

Cynicism runs wild. "The **keypuncher** types cards to feed the computer. Other people tell her exactly what to type. I say 'her' because most keypunchers are poorly paid women. The **operator** is a baby-sitter; he takes care of the computer. When it's hungry, he feeds it cards. When it spits up answers, he hands them to hu-

mans. If it throws a fit, he tries to calm it down; if he can't, he calls the **technician**, a repairman. The **programmer** translates English into language the computer can understand. A survey of 81 programmers at 14 computer centers shows that the average programmer has programmed only two years, took only two years of college, handled only two applications, knows only two computer languages, is introverted, sloppy and inflexible, feels he's in over his head, and is undermanaged.

Part 3 knocks off computer languages, with special emphasis on the ones programmers use most often: FORTRAN and COBOL. The carefree lunacy continues here: the first FORTRAN program talks about chubby chipmunks and giggling goldfish; the first COBOL program announces that Billie and Bonnie burp. Both languages are disposed of quickly: after reading just two pages about FORTRAN and two pages about COBOL, you can write your own complete programs in both languages. When you finish the FORTRAN and COBOL chapters, you'll be something of an expert. The text surveys the versions of FORTRAN and COBOL that computers actually use, instead of limiting itself to either the outdated versions (FORTRAN IV, COBOL 68) or not-quite-here versions (FORTRAN 77, COBOL 74). The book ends by comparing a dozen other languages.

To complete the series, part 4 dips down to the assembly-language, nuts-and-bolts level, and throws in true but naughty comments about various manufacturers.

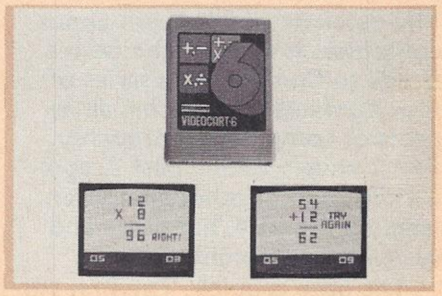
Part 1 costs $1.75; part 2, $2.50; part 3 will be available this May for about $3.50; part 4 in June for about $2.75. You can save 20% by ordering at least 20 (for example, 5 of each). Available from Russ Walter, 92 St. Botolph St., Boston, MA 02116.

Mighty Like a Rose

It's increasingly difficult to distinguish video game systems from computers. A videogame these days has a power supply, a microprocessor, a video driver, some memory, a player-controlled input channel, and programming capability. How soon will the games and computers merge?

Fairchild Camera and Instrument Corporation is hastening the process with developments in its Videocart™ game system, adding new games to its present repertoire on ROM plug-ins. *Spitfire,* for example, is an airplane dogfight for two players or for one player versus the console's microcomputer. *Spacewar* lets a couple of flying saucers fight it out. Less bloodthirsty (and presumably less fun) is the first non-game Videocart cartridge — *Math Quiz,* a drill that puts kids through their arithmetic paces. The Video Entertainment System lists at $169.95 retail, while the Videocart plug-ins list at $19.95.

Some of the ROM programs are quite complex and a good many useful computing programs might easily be packaged this way. Now, with a little memory expansion and an alphanumeric keyboard . . .



NEW GAME CARTRIDGES: Fairchild has added Videocarts 4, 5 and 6 — Spitfire, Space War and Math Quiz — to its library of game cartridges. Previously announced Videocarts feature Tic-Tac-Toe, Shooting Gallery, Doodle, Quadra-Doodle, Desert Fox and Black Jack. Tennis and Hockey are already in the systems' console. Suggested retail price for each Videocart is $19.95.

"MEET MY ACCOUNTANT!"

Complete Control.

Introducing IMSAI 8048 Single Board Control Computer.

Complete Control System
Intel developed the world's first single chip microcomputer. IMSAI has built the system to put it to work. Instantly!

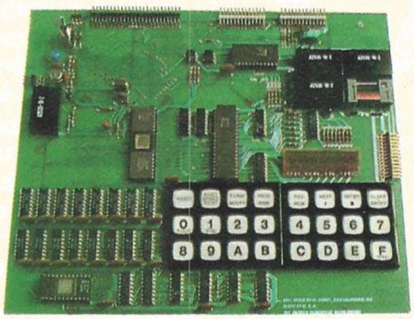
Presenting the IMSAI Single Board Control Computer. A complete programmable computer and hardware control system on an 8½" x 10" board.

Just plug in the power source (a 5V power supply or a 6V battery will do), and you can be controlling your home environment today!

The Chip Designed for Control
Intel designed the 8048/8748 single chip microcomputer with one thought in mind. Complete control. Everything you need is there: CPU, RAM, 1K ROM/EROM, I/O, timer/counter, interrupts . . . the works.

The Board That Implements It Instantly
There's a 24 pad hexadecimal keyboard and 9-digit LED hex display already on board. So you can start controlling the coffeepot and the sprinklers the minute you get the IMSAI home. Without adding anything but the connecting wires.

You can run almost any peripheral available. Not to mention the kids' electric train. The IMSAI Control Computer is



RS232 compatible. There are 12 quasi-bidirectional I/O lines with handshaking, and 14 more regular I/O lines, 5 heavy duty relays, and Teletype and audio cassette interfaces. All on one board.

There's already an extra 1K of RAM on board, plus sockets for another 1K of RAM and 2K of ROM/EROM. Still need more memory? The IMSAI 8048 allows expansion up to 64K of RAM off board.

Ultimately, the only limit to this system is your imagination.

Now, that's control.

For instant control, use the coupon provided.

CIRCLE 47

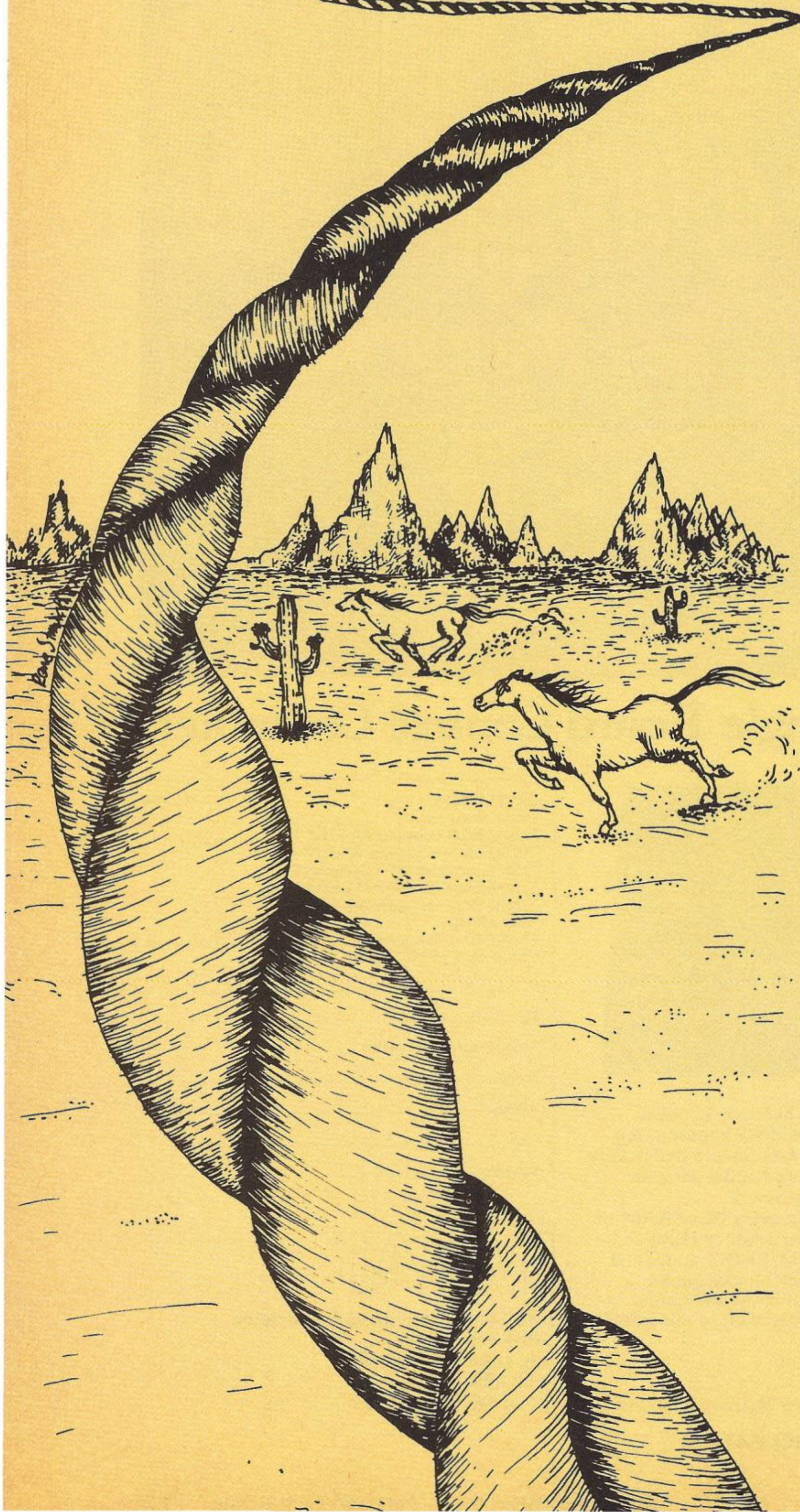
GENTLEMEN: PC-56
I want instant control!
☐ Send ROM Computer Kit $249*
☐ Send EROM Computer Kit $399
☐ Send assembled ROM Computer $299*
☐ Send assembled EROM Computer $499
☐ Send 5V power supply $99
*June delivery.
Check/MO enclosed. Amt. $____
Charge my: ☐ BAC ☐ M/C
#____ Exp. Date ____
Sig. ____
☐ Send more information.
Name ____
Address ____
City ____
State/Zip ____

IMSAI®

IMSAI Manufacturing Corporation
14860 Wicks Blvd.
San Leandro, CA 94577
(415) 483-2093
TWX 910-366-7287

CLUB ROUNDUP

by Louise Garcia



Denver Amateur Computer Society: "Interrupt"

Monthly meetings of the Denver Amateur Computer Society are designed to provide the opportunity to meet and exchange information and materials . . . each meeting consisting of three sections: Business Section, Lecture and Informal Exchange. The latter section allows members to visit displays, talk with old friends, new acquaintances and swap material or programs with others.

Editor Jim Clark reveals how he interfaced his Selectric 735 I/O writer to his Digital Group System. There's a hardware description, software description. Clark promises the interface to be easily wire wrapped or even wired point to point – says there's more info to follow as it becomes available.

Another purposeful article acquaints readers with basic operation of switching regulators and illustrates application in an uninterruptable power supply. Fred Eifert (author) contends that, until recently, the use of switching regulators has been virtually ignored. "Switch on to the Switching Regulator" considers the operation of the conventional series pass regulator and shows the advantages of the switching regulator in saving power. This power saving can be used to power more circuits or just save energy. Eifert concludes, "the switching regulator will give amazing results if constructed with normal care and consideration." You'll be glad you switched!

Subscriptions to INTERRUPT – $9.00 per year. Mail your order to DACS, P.O. Box 6338, Denver, CO 80206.

Santa Barbara Computer Association

Glen A. McComb, newsletter editor, proudly announces the premier issue – says their goals are simple – "provide interesting reading, both informative and educational; coordinate meetings and provide interesting formats; . . .review hardware and software . . ."

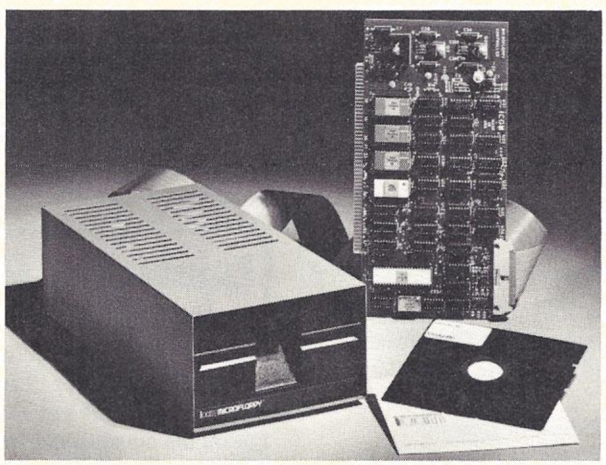
Grant Runyan's article, "Build This Logic Probe" presents a schematic (with some changes for the better) of a

Guess who just bought our new Microfloppy™ for $1095 including software?

Plug-compatible for $1095
iCOM's brand new Microfloppy™ is specifically designed for the personal computer user. It's smaller, using the new 5¼-inch diskette, and is priced at only $1095 including disk drive, power supply, cabinet, controller/interface card, all cables and connectors, plus, on diskette, iCOM's new FDOS-M software, including assembler, editor, and file manager. And for a limited time,* you also get an 8K Disk BASIC package at no extra cost.

Best of all, iCOM's Microfloppy™ is fully assembled and tested. It's 100% compatible with Altair, IMSAI, Poly 88 and other microcomputers using the Altair bus format.

*Offer good through June 15, 1977



Much faster than a cassette
This compact, economical Floppy Disk System is 10 to 15 times faster than a cassette. Programs can be loaded in seconds, a source program of several thousand characters can be corrected and a new file stored on diskette in less than a minute. Any of up to 175 files can be reached in 1.5 seconds.

iCOM — the leader
We've been building Floppy Disk Systems for microcomputers for nearly three years. Thousands of our systems are operating perfectly in the field. Many major computer manufacturers have incorporated our Floppy Disks into their systems. We deliver, and we'll be around to give you service whenever you may need it. You see, we're a division of Pertec Computer Corp., one of the largest and most respected manufacturers of peripherals, microsystems, data entry products, and data processing systems.

So you don't need to risk getting burned by a fly-by-night company. Order an iCOM® Microfloppy™ and be sure of getting the best.

See iCOM's Microfloppy™ now at your local computer store. Or contact us for the name of your dealer.

icom® MICROPERIPHERALS®

a division of Pertec Computer Corporation

6741 Variel Ave., Canoga Park, CA 91303 U.S.A.
(213) 348-1391 TWX 910-494-2788

SEE iCOM AT NCC – DALLAS – BOOTH 1415

CIRCLE 51

CLUB ROUNDUP

probe he built in a translucent disposable hypodermic syringe. The author believes that if only one test instrument can be afforded in working with digital hardware, it should be a logic test probe – and *this* one clips onto the 5 volt supply for its power!

Have you ever been "took"? Read the extraction from a letter written by CHEAP, INC., College Park, Maryland, which appears to be a parts house looking to do a decent job. This is a neat little discourse on the quality or lack of it in chips sold in bulk at bargain prices from various manufacturers. "Beware of Bargain Chips" describes some of the inadequacies in the surplus "bargains" and pointedly helps the reader determine what are the better investments. Any inquiries, comments and correspondence? Let Glenn McComb know at P.O. Box 2314, Goleta, CA 93018.

Association For Computing Machinery

The 1977 ACM roster of members is now available. The roster, which is cross-listed alphabetically and geographically, contains more than 35,000 ACM members. May be ordered for $25.00 to non-members by sending your money to ACM, Order Dept., P.O. Box 12105, Church Street Station, New York, N.Y. 10249.

Midwest Affiliation of Computer Clubs

Bargain hunters who are attracted to buying, selling and trading computing gear can find a real treat at Computerfest 77 opening June 10 at the Bond Court Hotel in Cleveland.

Sponsored by the Midwest Affiliation of Computer Clubs in Brecksville Ohio, Computerfest 77 is an effort to pool hobbyists, buyers, sellers, manufacturers. MACC (nonprofit) President, Gary Coleman tells us the ticket price is a fixed $2.00 – including everything from activities to exhibits to seminars and instruction. Contact Gary Coleman at MACC, P.O. Box 83, Cleveland, Ohio 44141.

Southern New England Computer Society

"Yankee Bits" reports that its second meeting of the year focused on computer hobby magazines (hmm-mm-mm) discussing what each had to offer and how to use each to the members' benefit. Sure would like to hear the fruits of *that* meeting.

South Florida Computer Group

Eric Whiteside of the Miami Group has started a series of software courses with an hour workshop on basic software logic. The course is taught at 7:00 PM, an hour before the meeting. Hardware workshop will begin at a later date.

The I/O Newsletter consists of pages chock full of infor – new product news, diagrams, charts, tables, meeting communiques, announcements, reports and hobbyist computer knowledge that's readable. If you'd like to be on the list of subscribers, write to Bruce Cameron, P.O. Box 236188, Miami, Florida 33123.

Central Texas Computer Association

"The President Speaks" (Roy McCoy) explains the membership growth – even a foreign subscriber or two . . . desperately in need of a Public Relations person.

In a brief, instructive article by Ron Parsons "SC/MP" Addressing" he describes switching two 1K independent blocks of memory to different logical addresses without soldering or unsoldering. His method can be modified for processors other than the SC/MP.

Lastique Publishing Co.

Is offering the Directory of U.S. Computer Clubs containing 5 cross-indexed sections. Might be a real aid in reaching persons of the same interest and coming to a meeting of the minds on problems. Another asset – the cost ($35.00) includes two up-dates. Write Dennis F. Lastique, P.O. Box 1691, Austin, Texas 78767 to order or for a listing form.

Cleveland Digital Group

"Open-house" is the format for CDG members and others who are interested in getting to know one another and learn what everyone's up to. Some who attend work with the 1004 line printer – others show off *their* computers. Open house begins at 7:00 PM, 15 minutes before the meeting!

Supplies, terminals, drives, chips and kits are available on group purchase and club discount – lots of good buys for hobbyists who pay in advance with shipping costs. Write or call Bob Uleski, 209 East 33 Street, Lorain, Ohio 44055, (216) 245-7388.

CLASSIFIEDS

Personal Computing Classified Rates

Individual: $1 per word; address set free; first two words set in all capital letters for free. If you want **bold** type, add $1 for each word set bold. Add $5 if you want replies sent to us and forwarded to you.

Company: 1X: $60/inch; 3X: $55/inch; 6X: $50/inch. Display ads accepted only if camera-ready materials submitted. Agency commission paid on camera-ready ads and after the first insertion of a publisher-set ad if subsequent insertions run without changes. Publisher sets ads in format and attempts to fit copy into space requested.

Advertisement accepted, however, with understanding that space may be larger than requested. Minimum billing size one inch; ads billed in half-inch increments thereafter. Frequency rate applies to publisher-set ads only if they run without changes in subsequent issues.

To place an ad, send your copy to Classified Ad Dept., **Personal Computing,** *167 Corey Rd., Brookline, MA 02146.*

Personal Computing

ADVERTISERS' INDEX

ADVERTISING SALES OFFICES

Iortheast: Irwin Stern, 167 Corey Rd., Brookline, MA 02146; 617) 232-5470; Arthur Daks, Benwill Publishing Corp., 299 Madison Ave., N.Y., N.Y. 10017; (201) 661-0360 ● **Mid-Atlantic:** Robert Poggi, 69 So. Central Ave., Valley Stream, NY 11580; (516) 72-9550 ● **Mid-West**; Ralph Petersen, 1 Wheaton Ctr. No. 1706, Vheaton, IL 60187; (312) 653-2480 ● **Southwest:** David Bunnell, 01 Louisiana, S.E. No. G, Albuquerque, NM 87108; (505) 266-173 ● **West:** Yuri Spiro & Carol Stagg, 823 Enchanted Way, Pacific Palisades, CA 90272; (213) 454-0624

If you're moving...

Please allow at least six weeks for us to put your new address into our subscription computer. Send your new address and the mailing label with your old address to: Circulation Dept., Personal Computing, 167 Corey Rd., Brookline, MA 02146.

Or use this convenient form.

Please attach here the mailing label with your old address

Write in here your new address.

NAME ______________________

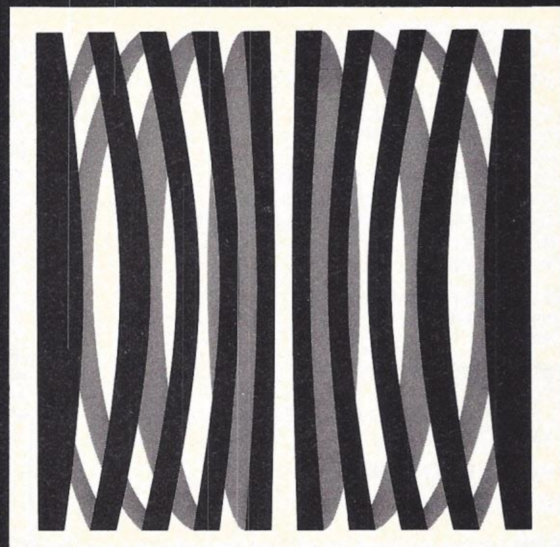
ADDRESS ______________________

CITY __________ STATE __________ ZIP ______

FEATURED ARTIST

Aldo Giorgini explains carefully that he never, as a child, seriously considered a future in art. Instead, he worked diligently toward a career in engineering and it was not until the years 1966-67, when he was a Postdoctoral Fellow at the National Center for Atmospheric Research , that he discovered the usefulness of the computer as a tool for producing "art." Since then his diligence as an artist in the computer and conventional media has brought him international acclaim.

Of course, this simple description of Giorgini's background omits an important fact or two. At age ten, he was apprenticed in Italy to Carlo Ingeneri, a now-well-known painter and sculptor who waived his fees when the lad's work began to pay his way. Years later, when the family returned to Italy after the War, Ambrogio Casati, another painter and sculptor of note, offered Giorgini an apprenticeship. Though the young man spent an average of three hours a day at the studio, he did not look upon art as his chosen career. Ah, that's the key, the **choice** was his. Giorgini is not just a highly trained technologist dabbling in art because he discovered that his equipment produces attractive patterns, but a skilled man disciplined in the fields of both art and technology.



In ARTIST AND COMPUTER (a Creative Computing Book published by Harmony Books, a Giorgini commented usefully on his computer art work. To the question, "Could your work be done without the aid of a computer?" he replied: "Yes, in the fashion analagous to the one of carving marble with a sponge. Since all FORTRAN instructions could be performed by other techniques, there is no doubt that one could execute the same by calculating and drawing by hand on a Cartesian plane."

"The motive substratum of my artistic activity is constituted by the fascination that natural forms have always exerted on me: from the extremely complex organic forms, rich of life-like attributes to the geometric forms of crystalline formations and to the forms of the invisible fields around us. My mental projection of the visual elements that 'describe' natural forms is constituted not by their 'substance,' their being objects, but by the surfaces that delimit such forms. In other words, I am not interested in recognizable individual objects, but in recognizable forms, be they organic, straight-line geometric or free-flowing geometric." However, the artist perceives his own work, the casual viewer finds Giorgini's product extraordinarily attractive.

Giorgini describes the stunning flag opposite in these terms: "A rectangular surface with markings representing the American flag is sinusoidally deformed in a direction orthogonal to its former plane. This surface is further deformed by flaring the upper and lower border until the surface touches itself. The surface is opaque."

Whatever. It's a grand old flag.

SWTPC 6800

- ★ COMPLETE WITH 2K OF MEMORY
- ★ SERIAL INTERFACE
- ★ STANDARD ROM MONITOR (Motorola MC 6830L7)
- ★ 4K AND 8K BASIC AVAILABLE

Always the best value in hardware and now an outstanding selection of software too. What more could you want in a computer system? For less than four hundred dollars you get everything you need—ready to connect to a terminal and go to work. No surprises no funny business, just good reliable hardware in a very practical system that may be expanded to meet almost any later need.

Memory and interfaces are not extra cost items in our system. A standard Motorola MIKBUG® ROM monitor makes the system completely compatible with Motorola software and eliminates any need for console switches and light. Data may be entered from the terminal in convenient hexidecimal form. The power supply is adequate to operate a fully expanded system with up to 24K of memory and up to eight (8) interfaces—simultaneously.

® Motorola

See the 6800 and our peripheral equipment at your nearest dealer, or write for a complete description.

MP-68 COMPUTER KIT—with serial interface, 2k of memory and ROM monitor **$395.00 ppd**

Southwest Technical Products Corporation, 219 W. Rhapsody, San Antonio, Texas 78216